



UltraLite® .NET Programming

Published: March 2007

Copyright and trademarks

Copyright © 2007 iAnywhere Solutions, Inc. Portions copyright © 2007 Sybase, Inc. All rights reserved.

iAnywhere Solutions, Inc. is a subsidiary of Sybase, Inc.

iAnywhere grants you permission to use this document for your own informational, educational, and other non-commercial purposes; provided that (1) you include this and all other copyright and proprietary notices in the document in all copies; (2) you do not attempt to "pass-off" the document as your own; and (3) you do not modify the document. You may not publish or distribute the document or any portion thereof without the express prior written consent of iAnywhere.

This document is not a commitment on the part of iAnywhere to do or refrain from any activity, and iAnywhere may change the content of this document at its sole discretion without notice. Except as otherwise provided in a written agreement between you and iAnywhere, this document is provided "as is", and iAnywhere assumes no liability for its use or any inaccuracies it may contain.

iAnywhere®, Sybase®, and the marks listed at <http://www.iAnywhere.com/trademarks> are trademarks of Sybase, Inc. or its subsidiaries. ® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Contents

About This Manual	ix
SQL Anywhere documentation	x
Documentation conventions	xiii
Finding out more and providing feedback	xvii
 Introduction to UltraLite.NET	 1
UltraLite and .NET	2
System requirements and supported platforms	4
UltraLite.NET architecture	5
 SQL Anywhere Explorer for UltraLite	 7
Introduction to the SQL Anywhere Explorer	8
Using the SQL Anywhere Explorer in UltraLite projects	9
 Understanding UltraLite.NET Development	 13
Using SQL Anywhere tools in Visual Studio .NET	14
Connecting to a database	15
Encryption and obfuscation	17
Accessing and manipulating data using SQL	18
Accessing and manipulating data with the Table API	22
Managing transactions	28
Accessing schema information	29
Handling errors	30
Authenticating users	31
Synchronization in UltraLite applications	32
 Tutorial: Build an UltraLite.NET Application	 35
Introduction to UltraLite.NET development tutorial	36
Lesson 1: Create a Visual Studio project	37
Lesson 2: Create an UltraLite database	40

Lesson 3: Connect to the database	41
Lesson 4: Insert, update, and delete data	43
Lesson 5: Build and deploy application	48
 UltraLite .NET 1.0 API Reference	51
ULActiveSyncListener interface	53
ULAuthStatusCode enumeration	56
ULCommand class	57
ULCommandBuilder class	82
ULConnection class	90
ULConnectionParms class	122
ULConnectionParms.UnusedEventHandler delegate	133
ULCreateParms class	134
ULCursorSchema class	144
ULDataAdapter class	153
ULDatabaseManager class	178
ULDatabaseSchema class	185
ULDataReader class	195
ULDateOrder enumeration	237
ULDbType enumeration	238
ULException class	242
ULFileTransfer class	246
ULFileTransferProgressData class	260
ULFileTransferProgressListener interface	263
ULIndexSchema class	265
ULInfoMessageEventArgs class	273
ULInfoMessageEventHandler delegate	276
ULParameter class	277
ULParameterCollection class	294
ULPublicationSchema class	312
ULResultSet class	316
ULResultSetSchema class	342
ULRowUpdatedEventArgs class	344
ULRowUpdatedEventHandler delegate	348
ULRowUpdatingEventArgs class	349

ULRowUpdatingEventHandler delegate	352
ULRuntimeType enumeration	353
ULServerSyncListener interface	354
ULSQLCode enumeration	357
ULStreamErrorCode enumeration	366
ULStreamErrorContext enumeration	381
ULStreamErrorID enumeration	382
ULStreamType enumeration	384
ULSyncParms class	385
ULSyncProgressData class	399
ULSyncProgressListener interface	409
ULSyncProgressState enumeration	411
ULSyncResult class	413
ULTable class	419
ULTableSchema class	441
ULTransaction class	456
 UltraLite .NET 2.0 API Reference	461
ULActiveSyncListener interface	463
ULAuthStatusCode enumeration	467
ULBulkCopy class	468
ULBulkCopyColumnMapping class	480
ULBulkCopyColumnMappingCollection class	487
ULBulkCopyOptions enumeration	496
ULCommand class	497
ULCommandBuilder class	532
ULConnection class	542
ULConnectionParms class	578
ULConnectionParms.UnusedEventHandler delegate	589
ULConnectionStringBuilder class	590
ULCreateParms class	607
ULCursorSchema class	618
ULDataAdapter class	627
ULDatabseManager class	639
ULDatabseSchema class	646

ULDataReader class	656
ULDateOrder enumeration	694
ULDbType enumeration	695
ULException class	699
ULFactory class	703
ULFileTransfer class	709
ULFileTransferProgressData class	724
ULFileTransferProgressListener interface	727
ULIndexSchema class	729
ULInfoMessageEventArgs class	738
ULInfoMessageEventHandler delegate	741
ULMetaDataCollectionNames class	742
ULParameter class	751
ULParameterCollection class	768
ULPublicationSchema class	788
ULResultSet class	792
ULResultSetSchema class	816
ULRowsCopiedEventArgs class	819
ULRowsCopiedEventHandler delegate	822
ULRowUpdatedEventArgs class	823
ULRowUpdatedEventHandler delegate	827
ULRowUpdatingEventArgs class	828
ULRowUpdatingEventHandler delegate	831
ULRuntimeType enumeration	832
ULServerSyncListener interface	833
ULSQLCode enumeration	836
ULStreamErrorCode enumeration	846
ULStreamErrorContext enumeration	864
ULStreamErrorID enumeration	865
ULStreamType enumeration	867
ULSyncParms class	868
ULSyncProgressData class	883
ULSyncProgressListener interface	894
ULSyncProgressState enumeration	896
ULSyncResult class	898

ULTable class	904
ULTableSchema class	925
ULTransaction class	939
Index	943

About This Manual

Subject

This manual describes UltraLite.NET. With UltraLite.NET you can develop and deploy database applications to computers, or handheld, mobile, or embedded devices.

Audience

This manual is intended for .NET application developers who want to take advantage of the performance, resource efficiency, robustness, and security of an UltraLite relational database for data storage and synchronization.

SQL Anywhere documentation

This book is part of the SQL Anywhere documentation set. This section describes the books in the documentation set and how you can use them.

The SQL Anywhere documentation

The complete SQL Anywhere documentation is available in two forms: an online form that combines all books, and as separate PDF files for each book. Both forms of the documentation contain identical information and consist of the following books:

- ◆ **SQL Anywhere 10 - Introduction** This book introduces SQL Anywhere 10—a product that provides data management and data exchange technologies, enabling the rapid development of database-powered applications for server, desktop, mobile, and remote office environments.
- ◆ **SQL Anywhere 10 - Changes and Upgrading** This book describes new features in SQL Anywhere 10 and in previous versions of the software, as well as upgrade instructions.
- ◆ **SQL Anywhere Server - Database Administration** This book covers material related to running, managing, and configuring SQL Anywhere databases. It describes database connections, the database server, database files, backup procedures, security, high availability, and replication with Replication Server, as well as administration utilities and options.
- ◆ **SQL Anywhere Server - SQL Usage** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.
- ◆ **SQL Anywhere Server - SQL Reference** This book provides a complete reference for the SQL language used by SQL Anywhere. It also describes the SQL Anywhere system views and procedures.
- ◆ **SQL Anywhere Server - Programming** This book describes how to build and deploy database applications using the C, C++, and Java programming languages, as well as Visual Studio .NET. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by these tools.
- ◆ **SQL Anywhere 10 - Error Messages** This book provides a complete listing of SQL Anywhere error messages together with diagnostic information.
- ◆ **MobiLink - Getting Started** This manual introduces MobiLink, a session-based relational-database synchronization system. MobiLink technology allows two-way replication and is well suited to mobile computing environments.
- ◆ **MobiLink - Server Administration** This manual describes how to set up and administer MobiLink server-side utilities and functionality.
- ◆ **MobiLink - Client Administration** This manual describes how to set up, configure, and synchronize MobiLink clients. MobiLink clients can be SQL Anywhere or UltraLite databases.
- ◆ **MobiLink - Server-Initiated Synchronization** This manual describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization or other remote actions from the consolidated database.

- ◆ **QAnywhere** This manual describes QAnywhere, which is a messaging platform for mobile and wireless clients as well as traditional desktop and laptop clients.
- ◆ **SQL Remote** This book describes the SQL Remote data replication system for mobile computing, which enables sharing of data between a SQL Anywhere consolidated database and many SQL Anywhere remote databases using an indirect link such as email or file transfer.
- ◆ **SQL Anywhere 10 - Context-Sensitive Help** This manual contains the context-sensitive help for the Connect dialog, the Query Editor, the MobiLink Monitor, MobiLink Model mode, the SQL Anywhere Console utility, the Index Consultant, and Interactive SQL.
- ◆ **UltraLite - Database Management and Reference** This manual introduces the UltraLite database system for small devices.
- ◆ **UltraLite - AppForge Programming** This manual describes UltraLite for AppForge. With UltraLite for AppForge you can develop and deploy database applications to handheld, mobile, or embedded devices, running Palm OS, Symbian OS, or Windows CE.
- ◆ **UltraLite - .NET Programming** This manual describes UltraLite.NET. With UltraLite.NET you can develop and deploy database applications to computers, or handheld, mobile, or embedded devices.
- ◆ **UltraLite - M-Business Anywhere Programming** This manual describes UltraLite for M-Business Anywhere. With UltraLite for M-Business Anywhere you can develop and deploy web-based database applications to handheld, mobile, or embedded devices, running Palm OS, Windows CE, or Windows XP.
- ◆ **UltraLite - C and C++ Programming** This manual describes UltraLite C and C++ programming interfaces. With UltraLite, you can develop and deploy database applications to handheld, mobile, or embedded devices.

Documentation formats

SQL Anywhere provides documentation in the following formats:

- ◆ **Online documentation** The online documentation contains the complete SQL Anywhere documentation, including the books and the context-sensitive help for SQL Anywhere tools. The online documentation is updated with each maintenance release of the product, and is the most complete and up-to-date source of documentation.

To access the online documentation on Windows operating systems, choose Start ► Programs ► SQL Anywhere 10 ► Online Books. You can navigate the online documentation using the HTML Help table of contents, index, and search facility in the left pane, as well as using the links and menus in the right pane.

To access the online documentation on Unix operating systems, see the HTML documentation under your SQL Anywhere installation or on your installation CD.

- ◆ **PDF files** The complete set of SQL Anywhere books is provided as a set of Adobe Portable Document Format (pdf) files, viewable with Adobe Reader.

On Windows, the PDF books are accessible from the online documentation via the PDF link at the top of each page, or from the Windows Start menu (Start ► Programs ► SQL Anywhere 10 ► Online Books - PDF Format).

On Unix, the PDF books are available on your installation CD.

Documentation conventions

This section lists the typographic and graphical conventions used in this documentation.

Syntax conventions

The following conventions are used in the SQL syntax descriptions:

- ◆ **Keywords** All SQL keywords appear in uppercase, like the words `ALTER TABLE` in the following example:

`ALTER TABLE [ owner.]table-name`

- ◆ **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown like the words *owner* and *table-name* in the following example:

`ALTER TABLE [ owner.]table-name`

- ◆ **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots), like *column-constraint* in the following example:

`ADD column-definition [ column-constraint, ... ]`

One or more list elements are allowed. In this example, if more than one is specified, they must be separated by commas.

- ◆ **Optional portions** Optional portions of a statement are enclosed by square brackets.

`RELEASE SAVEPOINT [ savepoint-name ]`

These square brackets indicate that the *savepoint-name* is optional. The square brackets should not be typed.

- ◆ **Options** When none or only one of a list of items can be chosen, vertical bars separate the items and the list is enclosed in square brackets.

`[ ASC | DESC ]`

For example, you can choose one of `ASC`, `DESC`, or neither. The square brackets should not be typed.

- ◆ **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces and a bar is used to separate the options.

`[ QUOTES { ON | OFF } ]`

If the `QUOTES` option is used, one of `ON` or `OFF` must be provided. The brackets and braces should not be typed.

Operating system conventions

- ◆ **Windows** The Microsoft Windows family of operating systems for desktop and laptop computers. The Windows family includes Windows Vista and Windows XP.

- ◆ **Windows CE** Platforms built from the Microsoft Windows CE modular operating system, including the Windows Mobile and Windows Embedded CE platforms.

Windows Mobile is built on Windows CE. It provides a Windows user interface and additional functionality, such as small versions of applications like Word and Excel. Windows Mobile is most commonly seen on mobile devices.

Limitations or variations in SQL Anywhere are commonly based on the underlying operating system (Windows CE), and seldom on the particular variant used (Windows Mobile).

- ◆ **Unix** Unless specified, Unix refers to both Linux and Unix platforms.

File name conventions

The documentation generally adopts Windows conventions when describing operating system dependent tasks and features such as paths and file names. In most cases, there is a simple transformation to the syntax used on other operating systems.

- ◆ **Directories and path names** The documentation typically lists directory paths using Windows conventions, including colons for drives and backslashes as a directory separator. For example,

`MobiLink\redirector`

On Unix, Linux, and Mac OS X, you should use forward slashes instead. For example,

`MobiLink/redirector`

If SQL Anywhere is used in a multi-platform environment you must be aware of path name differences between platforms.

- ◆ **Executable files** The documentation shows executable file names using Windows conventions, with the suffix `.exe`. On Unix, Linux, and Mac OS X, executable file names have no suffix. On NetWare, executable file names use the suffix `.nlm`.

For example, on Windows, the network database server is `dbsrv10.exe`. On Unix, Linux, and Mac OS X, it is `dbsrv10`. On NetWare, it is `dbsrv10.nlm`.

- ◆ **install-dir** The installation process allows you to choose where to install SQL Anywhere, and the documentation refers to this location using the convention *install-dir*.

After installation is complete, the environment variable `SQLANY10` specifies the location of the installation directory containing the SQL Anywhere components (*install-dir*). `SQLANYSH10` specifies the location of the directory containing components shared by SQL Anywhere with other Sybase applications.

For more information on the default location of *install-dir*, by operating system, see [“SQLANY10 environment variable” \[SQL Anywhere Server - Database Administration\]](#).

- ◆ **samples-dir** The installation process allows you to choose where to install the samples that are included with SQL Anywhere, and the documentation refers to this location using the convention *samples-dir*.

After installation is complete, the environment variable `SQLANYSAMP10` specifies the location of the directory containing the samples (*samples-dir*). From the Windows Start menu, choosing Programs ► SQL Anywhere 10 ► Sample Applications and Projects opens a Windows Explorer window in this directory.

For more information on the default location of *samples-dir*, by operating system, see [“Samples directory” \[SQL Anywhere Server - Database Administration\]](#).

- ◆ **Environment variables** The documentation refers to setting environment variables. On Windows, environment variables are referred to using the syntax `%envvar%`. On Unix, Linux, and Mac OS X, environment variables are referred to using the syntax `$envvar` or `${envvar}`.

Unix, Linux, and Mac OS X environment variables are stored in shell and login startup files, such as `.cshrc` or `.tcshrc`.

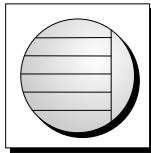
Graphic icons

The following icons are used in this documentation.

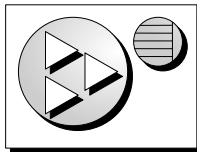
- ◆ A client application.



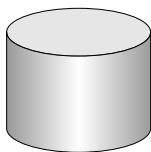
- ◆ A database server, such as SQL Anywhere.



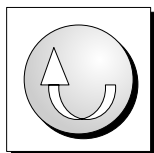
- ◆ An UltraLite application.



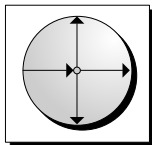
- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ◆ Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink server and the SQL Remote Message Agent.



- ◆ A Sybase Replication Server



- ◆ A programming interface.



Finding out more and providing feedback

Finding out more

Additional information and resources, including a code exchange, are available at the iAnywhere Developer Network at <http://www.ianywhere.com/developer/>.

If you have questions or need help, you can post messages to the Sybase iAnywhere newsgroups listed below.

When you write to one of these newsgroups, always provide detailed information about your problem, including the build number of your version of SQL Anywhere. You can find this information by entering **dbeng10 -v** at a command prompt.

The newsgroups are located on the *forums.sybase.com* news server. The newsgroups include the following:

- ◆ [sybase.public.sqlanywhere.general](#)
- ◆ [sybase.public.sqlanywhere.linux](#)
- ◆ [sybase.public.sqlanywhere.mobilink](#)
- ◆ [sybase.public.sqlanywhere.product_futures_discussion](#)
- ◆ [sybase.public.sqlanywhere.replication](#)
- ◆ [sybase.public.sqlanywhere.ultralite](#)
- ◆ [ianywhere.public.sqlanywhere.qanywhere](#)

Newsgroup disclaimer

iAnywhere Solutions has no obligation to provide solutions, information, or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and ensure its operation and availability.

iAnywhere Technical Advisors as well as other staff assist on the newsgroup service when they have time available. They offer their help on a volunteer basis and may not be available on a regular basis to provide solutions and information. Their ability to help is based on their workload.

Feedback

We would like to receive your opinions, suggestions, and feedback on this documentation.

You can email comments and suggestions to the SQL Anywhere documentation team at iasdoc@ianywhere.com. Although we do not reply to emails sent to that address, we read all suggestions with interest.

In addition, you can provide feedback on the documentation and the software through the newsgroups listed above.

CHAPTER 1

Introduction to UltraLite.NET

Contents

UltraLite and .NET 2

System requirements and supported platforms 4

UltraLite.NET architecture 5

UltraLite and .NET

The .NET Compact Framework is the Microsoft .NET runtime component for Windows CE. It supports several programming languages. You can use either Visual Basic.NET or C# to build applications using UltraLite.NET.

UltraLite.NET applications can be deployed to Windows CE and Windows. If you are deploying to Windows CE, UltraLite.NET requires the .NET Compact Framework. If deploying to Windows, it requires the .NET Framework. UltraLite.NET also supports ActiveSync synchronization.

UltraLite.NET provides the following namespace:

- ◆ **iAnywhere.Data.UltraLite** This namespace provides an ADO.NET interface to UltraLite. It has the advantage of being built on an industry-standard model and providing a migration path to the SQL Anywhere ADO.NET interface, which is very similar.

iAnywhere.Data.UltraLite namespace (.NET 1.0)

The iAnywhere.Data.UltraLite namespace contains the classes, interfaces, and enumerations of the UltraLite.NET Data Provider for ADO.NET. UltraLite.NET allows you to write C# or Visual Basic .NET code to develop UltraLite database applications using the ADO.NET standard and provides a migration path to SQL Anywhere.

If you are undecided as to whether to use UltraLite or SQL Anywhere, start with the iAnywhere.Data.UltraLite namespace and switch to SQL Anywhere (iAnywhere.Data.SQLAnywhere namespace) if you determine that more advanced SQL Anywhere features are required. If you may be moving to SQL Anywhere, avoid as much as possible the few UltraLite-only extensions that are in the iAnywhere.Data.UltraLite namespace.

UltraLite.NET extensions that are not available in the SQL Anywhere Data Provider for ADO.NET are denoted in this API reference with "UL Ext.:". To use the UltraLite Engine runtime of UltraLite.NET, set RuntimeType property to the appropriate value prior to using any other UltraLite.NET API.

Applications must open a connection to perform operations on a database. Connections are opened using the ULConnection class. The **iAnywhere.Data.UltraLite** assembly uses a satellite resource assembly called **iAnywhere.Data.UltraLite.resources**. The main assembly searches for this resource assembly by culture, using the following order: [CultureInfo.CurrentUICulture](#), then [CultureInfo.CurrentCulture](#), and finally culture "EN".

iAnywhere.Data.UltraLite namespace (.NET 2.0)

This chapter describes the API for the UltraLite .NET Data Provider for .NET Framework 2.0 and .NET Compact Framework 2.0.

UltraLite.NET extensions that are not available in the SQL Anywhere Data Provider for ADO.NET are denoted in this API reference with "UL Ext.:".

To use the UltraLite Engine runtime of UltraLite.NET, set [“RuntimeType property” on page 640](#) to the appropriate value prior to using any other UltraLite.NET API.

Applications must open a connection to perform operations on a database. Connections are opened using the [“ULConnection class” on page 542](#).

The **iAnywhere.Data.UltraLite** assembly uses a satellite resource assembly called **iAnywhere.Data.UltraLite.resources**. The main assembly searches for this resource assembly by culture, using the following order: [CultureInfo.CurrentUICulture](#), then [CultureInfo.CurrentCulture](#), and finally culture "EN".

Many of the properties and methods in this chapter are very similar to the .NET Framework Data Provider for OLE DB (System.Data.OleDb). You can find more information and examples in the Microsoft .NET Framework documentation.

System requirements and supported platforms

Development platforms

To develop applications using UltraLite.NET, you require the following:

- ◆ Microsoft Windows.
- ◆ Visual Studio.NET, Visual Studio.NET 2003, or Visual Studio 2005.
- ◆ For Windows CE devices, .NET Compact Framework version 1.0.5000 or later.

Target platforms

UltraLite.NET supports the following target platforms:

- ◆ Microsoft .NET Compact Framework 2.0 and .NET Framework 2.0.
- ◆ Microsoft .NET Compact Framework 1.0.3705 or later on Windows CE 3.0 and higher, with Pocket PC on ARM processors. The Pocket PC 2002 emulator is also supported.
- ◆ For Windows CE devices, Microsoft .NET Compact Framework version 1.0.5000 or later.
- ◆ Microsoft .NET Compact Framework version 1.0.5000 or later on Windows.

In addition to your application code and the .NET Compact Framework, you must deploy the following files to your Windows CE device or computer running Windows:

- ◆ **iAnywhere.Data.UltraLite.dll** An assembly containing the iAnywhere.Data.UltraLite ADO.NET namespace. This file is required only if your application uses the iAnywhere.Data.UltraLite namespace.
- ◆ **iAnywhere.Data.UltraLite.resources.dll** The resources needed by the iAnywhere.Data.UltraLite ADO.NET namespace. This file is required only if your application uses the iAnywhere.Data.UltraLite namespace.
- ◆ **ulnet10.dll** This file contains the UltraLite runtime. A separate version of the runtime is provided for each target platform.

For details about UltraLite supported platforms, see the UltraLite Deployment Option for SQL Anywhere [SQL Anywhere Supported Platforms and Engineering Support Status](#).

UltraLite.NET architecture

The UltraLite.NET namespace is named `iAnywhere.Data.UltraLite` (ADO.NET interface).

The following list describes some of the more commonly-used high level classes for the `iAnywhere.Data.UltraLite` ADO.NET namespace:

- ◆ **ULConnection** Each `ULConnection` object represents a connection to an UltraLite database. You can create one or more `ULConnection` objects.
- ◆ **ULTable** Each `ULTable` object provides access to the data in a single table.
- ◆ **ULCommand object** Each `ULCommand` object holds a SQL statement to be executed against the database.
- ◆ **ULDataReader object** Each `ULDataReader` object holds the result set for a single query.
- ◆ **ULSyncParms** You use the `ULSyncParms` object to synchronize your UltraLite database with a MobiLink server.

For more information about the ADO.NET interface, see [“UltraLite .NET 1.0 API Reference” on page 51](#).

CHAPTER 2

SQL Anywhere Explorer for UltraLite

Contents

Introduction to the SQL Anywhere Explorer 8

Using the SQL Anywhere Explorer in UltraLite projects 9

Introduction to the SQL Anywhere Explorer

The SQL Anywhere Explorer is a component that lets you connect to SQL Anywhere and UltraLite databases from Visual Studio .NET.

For information about using the SQL Anywhere Explorer with SQL Anywhere databases, see [“SQL Anywhere Explorer”](#) [*SQL Anywhere Server - Programming*].

Using the SQL Anywhere Explorer in UltraLite projects

In Visual Studio .NET 2003 and 2005, you can use the SQL Anywhere Explorer to create connections to UltraLite databases. Once you connect to a database, you can:

- ◆ browse tables and table data
- ◆ design programs to open connections with the UltraLite database, or to retrieve and manipulate data
- ◆ drag and drop database objects onto C# or Visual Basic code or forms so that the IDE automatically generates code that references the selected object

You can also open Sybase Central and Interactive SQL from Visual Studio .NET by choosing the corresponding command from the Tools menu.

Installation note

If you install UltraLite software on a Windows computer that already has Visual Studio installed, the installation process detects the presence of Visual Studio and performs the necessary integration steps. If you install Visual Studio after installing UltraLite, or install a new version of Visual Studio, the process to integrate UltraLite with Visual Studio must be performed manually as follows:

- ◆ Ensure Visual Studio is not running.
- ◆ For Visual Studio .NET 2003, run *install-dir\ultralite\UltraLite.NET\assembly\v1\installULNet.exe*.

For Visual Studio 2005, run *install-dir\ultralite\UltraLite.NET\assembly\v2\installULNet.exe*.

Working with UltraLite database connections in Visual Studio .NET

Use the SQL Anywhere Explorer to display the UltraLite database connections under the Data Connections node. You must create a data connection to view the data in the tables.

You can list database tables in the SQL Anywhere Explorer and expand individual tables to list their columns. The properties for an object selected in the SQL Anywhere Explorer window appear in the Visual Studio Properties pane.

◆ To add an UltraLite database connection in Visual Studio .NET

1. Open the SQL Anywhere Explorer by choosing View ► SQL Anywhere Explorer.
2. In the SQL Anywhere Explorer window, right-click Data Connections, and then choose Add Connection.

The Add Connection dialog appears.

3. Select UltraLite, and then click OK.

The Connection Properties dialog appears.

4. Enter the appropriate values to connect to your database.
5. Click OK.

A connection is made to the database, and the connection is added to the Data Connections list.

♦ **To remove an UltraLite database connection from Visual Studio .NET**

1. Open the SQL Anywhere Explorer by choosing View ► SQL Anywhere Explorer.
2. In the SQL Anywhere Explorer window, right-click the UltraLite data connections you want to remove, and then choose Delete.

The connection is removed from the SQL Anywhere Explorer window.

Configuring the SQL Anywhere Explorer

The Visual Studio options dialog includes settings that you can use to configure the SQL Anywhere Explorer. Some of the options are general options and some are specific to UltraLite usage only.

♦ **To access SQL Anywhere Explorer options**

1. From the Visual Studio Tools menu, choose Options.
The Options dialog appears.
2. In the left pane of the Options dialog, expand SQL Anywhere.
3. Click General to configure the SQL Anywhere Explorer general options as required.

Limit query results sent to Output window Specify the number of rows that appear in the Output window. The default value is 500.

Sort objects Choose to sort objects in the for UltraLite Explorer window by object name or by object owner name.

Generate UI code when dropping a table or view onto the Designer Generate the code for tables or views that you drag and drop onto the Windows Forms Designer.

Generate Insert, Update, and Delete commands for adapters Generate INSERT, UPDATE, and DELETE commands for the data adapter when you drag and drop a table or view onto a C# or Visual Basic document.

Generate table mappings for data adapters Generate table mappings for the data adapter when you drag and drop a table onto a C# or Visual Basic document.

4. Click UltraLite to configure the specific option for UltraLite.

Generate when dropping a table into code Generate code of a specific type for tables that you drag and drop into your application code. Choose from one of the following:

- ♦ **ULResultSet** represents an editable result set on which you can perform positioned updates and deletes.

- ◆ ULDataReader represents a read-only result set.
- ◆ ULTable represents code that allows you to store, remove, update, and read data from a table.
- ◆ ULDataAdapter gives you a method to work with table data offline.

Adding UltraLite database objects using the SQL Anywhere Explorer

In Visual Studio .NET, when you drag certain database objects from the SQL Anywhere Explorer and drop them onto Visual Studio .NET designers, the IDE automatically creates new components that reference the selected objects. You can configure the settings for drag and drop operations by choosing Tools ► Options in Visual Studio .NET.

The following table lists the UltraLite objects you can drag from the SQL Anywhere Explorer, and describes the components created when you drop them onto a Visual Studio .NET Forms Designer or Code Editor.

Item	Result
Data connection	Creates a data connection.
Table	Creates an adapter.

◆ To create a new UltraLite data component using the SQL Anywhere Explorer

1. Open the form or class that you want to add a data component to.
2. In the SQL Anywhere Explorer, select the UltraLite object you want to use.
3. Drag the object from the SQL Anywhere Explorer to the Forms Designer or Code Editor.

Working with UltraLite tables using the SQL Anywhere Explorer

The SQL Anywhere Explorer enables you to view the properties and data for UltraLite tables in database from within Visual Studio .NET.

◆ To view a table in Visual Studio .NET

1. Connect to an UltraLite database using the SQL Anywhere Explorer.
2. In the SQL Anywhere Explorer dialog, expand your UltraLite database, and then expand Tables.
3. Right-click a table, and then choose Retrieve Data.

The data in the selected table appears in the Output window in Visual Studio .NET.

CHAPTER 3

Understanding UltraLite.NET Development

Contents

Using SQL Anywhere tools in Visual Studio .NET	14
Connecting to a database	15
Encryption and obfuscation	17
Accessing and manipulating data using SQL	18
Accessing and manipulating data with the Table API	22
Managing transactions	28
Accessing schema information	29
Handling errors	30
Authenticating users	31
Synchronization in UltraLite applications	32

Using SQL Anywhere tools in Visual Studio .NET

Some SQL Anywhere tools are incorporated into Visual Studio .NET 2003 and 2005 to create an integrated database development environment for both SQL Anywhere and UltraLite databases.

- ◆ You can use the SQL Anywhere Explorer to view tables and data in those tables, as well as design programs to open connections to that database for data manipulation and retrieval. The SQL Anywhere Explorer is available from the View menu.
- ◆ You can open Sybase Central and Interactive SQL without having to leave Visual Studio .NET. Sybase Central and Interactive SQL are available from the Tools menu.

Note

For UltraLite development you cannot add adapters for views nor commands for stored procedures; these features are only available to SQL Anywhere databases.

See also

- ◆ [“Working with UltraLite database connections in Visual Studio .NET” on page 9](#)
- ◆ [“Adding UltraLite database objects using the SQL Anywhere Explorer” on page 11](#)

Connecting to a database

UltraLite applications must connect to a database before carrying out operations on the data in it. This section describes how to connect to an UltraLite database.

Using the ULConnection object

The following properties of the ULConnection object govern global application behavior.

- ◆ **Commit behavior** By default, UltraLite.NET applications are in AutoCommit mode. Each Insert, Update, or Delete statement is committed to the database immediately. You can use ULConnection.BeginTransaction to define the start of a transaction in your application.

See [“Managing transactions” on page 28](#).

- ◆ **User authentication** You can change the user ID and password for the application from the default values of DBA and sql by using methods to grant or revoke connection permissions. Each UltraLite database can define a maximum of four user IDs.

See [“Authenticating users” on page 31](#).

- ◆ **Synchronization** A set of objects governing synchronization is accessed from the Connection object.

See [“Synchronization in UltraLite applications” on page 32](#).

- ◆ **Tables** UltraLite tables are accessed using methods of the Connection object.

See [“Accessing and manipulating data with the Table API” on page 22](#).

- ◆ **Commands** A set of objects is provided to handle the execution of dynamic SQL statements and to navigate result sets.

See [“Accessing and manipulating data using SQL” on page 18](#).

See [“ULConnection class” on page 90](#).

Multi-threaded applications

Each ULConnection and all objects created from it should be used on a single thread. If your application requires multiple threads accessing the UltraLite database, each thread requires a separate connection. For example, if you design your application to perform synchronization in a separate thread, you must use a separate connection for the synchronization and you must open the connection from that thread.

◆ To connect to an UltraLite database

1. Declare a ULConnection object.

Most applications use a single connection to an UltraLite database and leave the connection open. Multiple connections are only required for multi-threaded data access. For this reason, it is often best to declare the ULConnection object as global to the application.

```
ULConnection conn;
```

2. Open a connection to an existing database.

UltraLite applications must deploy an initial database file or the application must include code to create the database file. The initial database file can be created by using Sybase Central or the command line utilities provided with UltraLite.

You can specify connection parameters either as a connection string or using the `ULConnectionParms` object. The following example illustrates using the `ULConnectionParms` object to connect to an UltraLite database named *mydata.udb*.

```
ULConnectionParms parms = new ULConnectionParms();  
parms.DatabaseOnDesktop = "mydata.udb";  
conn = new ULConnection( parms.ToString() );  
conn.Open();
```

Code samples in C#

The code samples in this chapter are in Microsoft C#. If you are using one of the other supported development tools, you must modify the instructions appropriately.

Encryption and obfuscation

By default, the data in a new UltraLite database is not encrypted. By specifying appropriate database creation parameters, the database may be created with strong encryption or with simple obfuscation. Obfuscation is a very weak form of keyless encryption that is only intended to prevent casual observation of the data in the database (with a low-level file examination utility for example).

Encryption

To create a database with strong encryption, specify an encryption key when creating the database using Sybase Central or specify the encryption key in the creation parameters if the database is created by calling `ULCreateDatabase` or using the `ulcreate` utility. To be effective, the encryption key should contain a combination of characters, numbers, and special symbols. Using a long encryption key reduces the chances of someone guessing the key.

Once a database is encrypted, the encryption key cannot be recovered. Access to the database is completely lost unless the proper encryption key is specified. Encryption keys should be treated as sensitive information and archived appropriately.

See “[UltraLite DBKEY connection parameter](#)” [*UltraLite - Database Management and Reference*].

You can change the encryption key for an existing UltraLite database by applying a new encryption key with the `Connection.ChangeEncryptionKey` method.

See “[ULConnection class](#)” on page 90 and “[ULConnectionParms class](#)” on page 122.

After the database is encrypted, connections to the database must specify the correct encryption key; otherwise, the connections fail.

Obfuscation

To obfuscate the database, specify **obfuscate=y** as a database creation parameter. For more information about database encryption and obfuscation parameters, see “[Choosing creation-time database properties for UltraLite](#)” [*UltraLite - Database Management and Reference*].

Accessing and manipulating data using SQL

UltraLite applications can access table data using SQL statements or the Table API. This section describes data access using SQL statements.

For information about using the Table API, see [“Accessing and manipulating data with the Table API” on page 22](#).

This section explains how to perform the following tasks using SQL:

- ◆ Inserting, deleting, and updating rows.
- ◆ Executing queries and retrieving rows to a result set.
- ◆ Scrolling through the rows of a result set.

This section does not describe the SQL language itself. For more information about SQL features, see [“SQL Statements” \[SQL Anywhere Server - SQL Reference\]](#).

Data manipulation: INSERT, UPDATE, and DELETE

With UltraLite, you can perform SQL data manipulation language operations. These operations are performed using the `ULCommand.ExecuteNonQuery` method.

See [“ULCommand class” on page 57](#)

Placeholders for parameters in SQL statements are indicated by the `?` character. For any INSERT, UPDATE, or DELETE, each `?` is referenced according to its ordinal position in the command's parameters collection. For example, the first `?` is referred to as 0, and the second as 1.

◆ To insert a row

1. Declare a `ULCommand`.

```
ULCommand cmd;
```

2. Assign a SQL statement to the `ULCommand` object.

```
cmd = conn.CreateCommand();  
cmd.Command =  
    "INSERT INTO MyTable(MyColumn) values (?);"
```

3. Assign input parameter values for the statement.

The following code shows a string parameter.

```
String newValue;  
// assign value  
cmd.Parameters.add("", newValue);
```

4. Execute the statement.

The return value indicates the number of rows affected by the statement.

```
int rowsInserted = cmd.ExecuteNonQuery();
```

5. If you are using explicit transactions, commit the change.

```
myTransaction.Commit();
```

◆ To update a row

1. Declare a ULCommand.

```
ULCommand cmd;
```

2. Assign a statement to the ULCommand object.

```
cmd = conn.CreateCommand();  
cmd.Command =  
    "UPDATE MyTable SET MyColumn1 = ? WHERE MyColumn2 = ?";
```

3. Assign input parameter values for the statement.

```
String newValue;  
String oldValue;  
// assign values  
cmd.Parameters.add("", newValue);  
cmd.Parameters.add("", oldValue);
```

4. Execute the statement.

```
int rowsUpdated = cmd.ExecuteNonQuery();
```

5. If you are using explicit transactions, commit the change.

```
myTransaction.Commit();
```

◆ To delete a row

1. Declare a ULCommand.

```
ULCommand cmd;
```

2. Assign a statement to the ULCommand object.

```
cmd = conn.CreateCommand();  
cmd.Command =  
    "DELETE FROM MyTable WHERE MyColumn = ?";
```

3. Assign input parameter values for the statement.

```
String deleteValue;  
// assign value  
cmd.Parameters.add("", deleteValue);
```

4. Execute the statement.

```
int rowsDeleted = cmd.ExecuteNonQuery();
```

5. If you are using explicit transactions, commit the change.

```
myTransaction.Commit();
```

Data retrieval: SELECT

The SELECT statement allows you to retrieve information from the database. This section describes how to execute a SELECT statement and how to handle the result set it returns.

◆ To execute a SELECT statement

1. Declare a ULCommand object, which holds the query.

```
ULCommand cmd;
```

2. Assign a statement to the object.

```
cmd = conn.CreateCommand();  
cmd.Command =  
    "SELECT MyColumn FROM MyTable";
```

3. Execute the statement.

Query results can be returned as one of several types of objects. In this example, a ULDataReader object is used. In the following code, the result of the SELECT statement contains a string, which is output to the console window.

```
ULDataReader customerNames = prepStmt.ExecuteReader();  
int fc = customerNames.GetFieldCount();  
while( customerNames.MoveNext() ) {  
    for ( int i = 0;  
          i < fc;  
          i++ ) {  
        System.Console.Write(  
            customerNames.GetString( i ) + " " );  
    }  
    System.Console.WriteLine();  
}
```

Navigating SQL result sets

You can navigate through a result set using methods associated with the ULDataReader object.

The result set object provides you with the following methods to navigate a result set:

- ◆ **MoveAfterLast** moves to a position after the last row.
- ◆ **MoveBeforeFirst** moves to a position before the first row.
- ◆ **MoveFirst** moves to the first row.
- ◆ **MoveLast** moves to the last row.
- ◆ **MoveNext** moves to the next row.
- ◆ **MovePrevious** moves to the previous row.
- ◆ **MoveRelative(offset)** moves a certain number of rows relative to the current row, as specified by the offset. Positive offset values move forward in the result set, relative to the current position of the cursor

in the result set, and negative offset values move backward in the result set. An offset value of zero does not move the cursor, but allows you to repopulate the row buffer.

Result set schema description

The `ULDataReader.GetSchemaTable` method and `ULDataReader.Schema` property allow you to retrieve information about a result set, such as column names, total number of columns, column scales, column sizes, and column SQL types.

Example

The following example demonstrates how to use the `ULDataReader.Schema` and `ResultSet.Schema` properties to display schema information in a console window.

```
for ( int i = 0;
      i < MyResultSet.Schema.GetColumnCount();
      i++ ) {
    System.Console.WriteLine(
        MyResultSet.Schema.GetColumnName(i) + " " +
        MyResultSet.Schema.GetColumnSQLType(i)
    );
}
```

Accessing and manipulating data with the Table API

UltraLite applications can access table data using SQL statements or by using the Table API. This section describes data access using the Table API.

For more information about SQL, see [“Accessing and manipulating data using SQL” on page 18](#).

This section explains how to perform the following tasks using the Table API:

- ◆ Scroll through the rows of a table.
- ◆ Access the values of the current row.
- ◆ Use find and lookup methods to locate rows in a table.
- ◆ Insert, delete, and update rows.

Navigating the rows of a table

UltraLite.NET provides you with a number of methods to navigate a table to perform a wide range of navigation tasks.

The table object provides you with the following methods to navigate a table.

- ◆ **MoveAfterLast** moves to a position after the last row.
- ◆ **MoveBeforeFirst** moves to a position before the first row.
- ◆ **MoveFirst** moves to the first row.
- ◆ **MoveLast** moves to the last row.
- ◆ **MoveNext** moves to the next row.
- ◆ **MovePrevious** moves to the previous row.
- ◆ **MoveRelative(offset)** moves a certain number of rows relative to the current row, as specified by the offset. Positive offset values move forward in the table, relative to the current position of the cursor in the table, and negative offset values move backward in the table. An offset value of zero does not move the cursor, but allows you to repopulate the row buffer.

Example

The following code opens the MyTable table and displays the value of the MyColumn column for each row.

```
ULTable t = conn.ExecuteTable( "MyTable" );
int colID = t.GetOrdinal( "MyColumn" );
while ( t.MoveNext() ){
    System.Console.WriteLine( t.GetString( colID ) );
}
```

You expose the rows of the table to the application when you open the table object. By default, the rows are ordered by primary key value, but you can specify an index when opening a table to access the rows in a particular order.

Example

The following code moves to the first row of the MyTable table as ordered by the ix_col index.

```
ULTable t = conn.ExecuteTable( "MyTable", "ix_col" );  
t.MoveFirst();
```

See [“ULTable class” on page 419](#) and [“ULTableSchema class” on page 441](#).

Using UltraLite modes

An UltraLite mode determines the purpose for which the values in the buffer will be used. UltraLite has the following four modes of operation, in addition to a default mode.

- ◆ **Insert mode** The data in the buffer is added to the table as a new row when the insert method is called.
- ◆ **Update mode** The data in the buffer replaces the current row when the update method is called.
- ◆ **Find mode** Used to locate a row whose value exactly matches the data in the buffer when one of the find methods is called.
- ◆ **Lookup mode** Used to locate a row whose value matches or is greater than the data in the buffer when one of the lookup methods is called.

Accessing the values of the current row

A Table object is always located at one of the following positions:

- ◆ Before the first row of the table.
- ◆ On a row of the table.
- ◆ After the last row of the table.

If the Table object is positioned on a row, you can use one of a set of methods appropriate for the data type to retrieve or modify the value of each column.

Retrieving column values

The Table object provides a set of methods for retrieving column values. These methods take the column ID as argument.

Examples

The following code retrieves the value of the lname column, which is a character string.

```
int lname = t.GetOrdinal( "lname" );  
string lastname = t.GetString( lname );
```

The following code retrieves the value of the cust_id column, which is an integer.

```
int cust_id = t.GetOrdinal( "cust_id" );  
int id = t.GetInt( cust_id );
```

Modifying column values

In addition to the methods for retrieving values, there are methods for setting values. These methods take the column ID and the value as arguments.

Example

For example, the following code sets the value of the lname column to Kaminski.

```
t.SetString( lname, "Kaminski" );
```

By assigning values to these properties you do not alter the value of the data in the database. You can assign values to the properties even if you are before the first row or after the last row of the table, but it is an error to try to access data when the current row is at one of these positions, for example, by assigning the property to a variable.

```
// This code is incorrect
t.MoveBeforeFirst();
id = t.GetInt( cust_id );
```

Casting values

The method you choose must match the data type you want to assign. UltraLite automatically casts database data types where they are compatible, so that you could use the getString method to fetch an integer value into a string variable, and so on. See [“Converting data types explicitly” \[UltraLite - Database Management and Reference\]](#).

Searching rows with find and lookup

UltraLite has several modes of operation for working with data. Two of these modes, the find and lookup modes, are used for searching. The Table object has methods corresponding to these modes for locating particular rows in a table.

Note

The columns searched using Find and Lookup methods must be in the index used to open the table.

- ◆ **Find methods** move to the first row that exactly matches specified search values, under the sort order specified when the Table object was opened. If the search values cannot be found, the application is positioned before the first or after the last row.
- ◆ **Lookup methods** move to the first row that matches or is greater than a specified search value, under the sort order specified when the Table object was opened.

◆ To search for a row

1. Enter find or lookup mode.

The mode is entered by calling a method on the table object. For example, the following code enters find mode.

```
t.FindBegin();
```

2. Set the search values.

You do this by setting values in the current row. Setting these values affects the buffer holding the current row only, not the database. For example, the following code sets the value in the buffer to Kaminski.

```
int lname = t.GetOrdinal( "lname" );  
t.SetString( lname, "Kaminski" );
```

3. Search for the row.

Use the appropriate method to carry out the search. For example, the following instruction looks for the first row that exactly matches the specified value in the current index.

For multi-column indexes, a value for the first column is always used, but you can omit the other columns.

```
tCustomer.FindFirst();
```

4. Search for the next instance of the row.

Use the appropriate method to carry out the search. For a find operation, FindNext locates the next instance of the parameters in the index. For a lookup, MoveNext locates the next instance.

See [“ULTable class” on page 419](#).

Updating rows

The following procedure describes how to update a row.

◆ To update a row

1. Move to the row you want to update.

You can move to a row by scrolling through the table or by searching the table using find or lookup methods.

2. Enter update mode.

For example, the following instruction enters update mode on table t.

```
t.BeginUpdate();
```

3. Set the new values for the row to be updated.

For example, the following instruction sets the id column in the buffer to 3.

```
t.SetInt( id , 3);
```

4. Execute the Update.

```
t.Update();
```

After the update operation, the current row is the row that has been updated. If you changed the value of a column in the index specified when the Table object was opened, the current row is undefined.

By default, UltraLite.NET operates in AutoCommit mode, so that the update is immediately applied to the row in permanent storage. If you have disabled AutoCommit mode, the update is not applied until you execute a commit operation. See [“Managing transactions” on page 28](#).

Caution

You cannot update the primary key value of a row: delete the row and add a new row instead.

Inserting rows

The steps to insert a row are very similar to those for updating rows, except that there is no need to locate a row in the table before carrying out the insert operation. The order of row insertion into the table has no significance.

Example

The following code inserts a new row.

```
t.InsertBegin();
t.SetInt( id, 3 );
t.SetString( lname, "Carlo" );
t.Insert();
```

If you do not set a value for one of the columns, and that column has a default, the default value is used. If the column has no default, one of the following entries is used:

- ◆ For nullable columns, NULL.
- ◆ For numeric columns that disallow NULL, zero.
- ◆ For character columns that disallow NULL, an empty string.
- ◆ To explicitly set a value to NULL, use the setDBNull method.

For update operations, an insert is applied to the database in permanent storage when a commit is carried out. In AutoCommit mode, a commit is carried out as part of the insert method.

Deleting rows

The steps to delete a row are simpler than to insert or update rows. There is no delete mode corresponding to the insert or update modes.

The following procedure deletes a row.

◆ To delete a row

1. Move to the row you want to delete.

2. Execute the Table.Delete method.

```
t.Delete();
```

Managing transactions

UltraLite provides transaction processing to ensure the integrity of the data in your database. A transaction is a logical unit of work. Either an entire transaction is executed, or none of the statements in the transaction are executed.

By default, UltraLite.NET operates in AutoCommit mode, so that each insert, update, or delete is executed as a separate transaction. Once the operation is complete, the change is made to the database.

To use multi-statement transactions, you must create a ULTransaction class object by calling ULConnection.BeginTransaction. For example, if your application transfers money between two accounts, both the deduction from the source account and the addition to the destination account must be completed as a distinct operation, otherwise both statements must not be completed.

If the connection has performed a valid transaction, you must execute ULTransaction.Commit statement to complete the transaction and commit the changes to your database. If the set of updates is to be abandoned, execute ULTransaction.Rollback statement to cancel and roll back all the operations of the transaction. Once a transaction has been committed or rolled back, the connection will revert to AutoCommit mode until a subsequent call to ULConnection.BeginTransaction.

For example, the following code fragment shows how to set up a transaction that involves multiple operations (avoiding the default autocommit behavior):

```
// Assuming an already open connection named conn
ULTransaction txn = conn.BeginTransaction(IsolationLevel.ReadUncommitted);
// Carry out transaction operations here
txn.Commit();
```

UltraLite Isolation Level

UltraLite supports only the IsolationLevel.ReadUncommitted member of the IsolationLevel enumeration.

Some SQL statements—especially statements that alter the structure of the database—cause any pending transactions to be committed. Examples of SQL statements that automatically commit transactions in progress are: CREATE TABLE and ALTER TABLE.

See [“ULConnection class” on page 90](#) and [“ULTransaction class” on page 456](#).

Accessing schema information

The objects in the table API represent tables, columns, indexes, and synchronization publications. Each object has a Schema property that provides access to information about the structure of that object.

You cannot modify the schema through the API. You can only retrieve information about the schema.

You can access the following schema objects and information:

- ◆ **DatabaseSchema** exposes the number and names of the tables in the database, as well as global properties such as the format of dates and times.

To obtain a ULDatabaseSchema object, access ULConnection.Schema.

See [“ULConnection class” on page 90](#).

- ◆ **TableSchema** The number and names of the columns and indexes for this table.

To obtain a ULTableSchema object, access ULTable.Schema.

- ◆ **IndexSchema** Information about the column in the index. As an index has no data directly associated with it there is no separate Index class, just a ULIndexSchema class.

To obtain a ULIndexSchema object, call the ULTableSchema.GetIndex, the ULTableSchema.GetOptimalIndex, or the ULTableSchema.GetPrimaryKey method.

- ◆ **PublicationSchema** A list of the tables and columns contained in a publication. Publications are comprised of schema only, there is no Publication object.

To obtain a ULPublicationSchema object, call the ULDatabaseSchema.GetPublicationSchema method.

See [“ULTableSchema class” on page 441](#).

Handling errors

You can use the standard .NET error-handling features to handle errors. Most UltraLite methods throw `ULException` errors. You can use `ULException.NativeError` to retrieve the `ULSQLCode` value assigned to this error. `ULException` has a `Message` property, which you can use to obtain a descriptive text of the error. `ULSQLCode` errors are negative numbers indicating the error type.

For a list of error codes, see [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

After synchronization, you can use the `SyncResult` property of the connection to obtain more detailed error information. For example, the following sample illustrates a possible technique for reporting errors that occur during synchronization:

```
public void Sync()
{
    try
    {
        _conn.Synchronize( this );
        _inSync = false;
    }
    catch( ULException uEx ){
        if( uEx.NativeError == ULSQLCode.SQLE_COMMUNICATIONS_ERROR )
        {
            MessageBox.Show(
                "StreamErrorCode = " +
                _conn.SyncResult.StreamErrorCode.ToString() + "\r\n"
                + "StreamErrorContext = " +
                _conn.SyncResult.StreamErrorContext + "\r\n"
                + "StreamErrorID = " + _conn.SyncResult.StreamErrorID + "\r\n"
                + "StreamErrorSystem = " + _conn.SyncResult.StreamErrorSystem
                + "\r\n"
            );
        }
        else
        {
            MessageBox.Show(uEx.Message);
        }
    }
    catch(System.Exception ex )
    {
        MessageBox.Show(ex.Message);
    }
}
```

See:

- ◆ [“ULSyncProgressListener interface” on page 409](#)
- ◆ [“ULSyncResult class” on page 413](#)

Authenticating users

New users must be added from an existing connection. As all UltraLite databases are created with a default user ID of DBA and password sql, you must first connect as this user.

You cannot directly change a user ID. Instead, add the new user ID and delete the existing user ID. UltraLite supports a maximum of four user IDs for each UltraLite database.

See [“Authenticating users” on page 31](#).

◆ To add a user or change a password for an existing user

1. Connect to the database using the user ID and password of an existing user.
2. Grant user access to the database with the desired password using the `ULConnection.GrantConnectTo` method.

This procedure is the same whether you are adding a new user or changing the password of an existing user.

See [“ULConnection class” on page 90](#).

◆ To delete an existing user

1. Connect to the database using the user ID and password of an existing user.
2. Delete an existing user using the `Connection.RevokeConnectFrom` method.

Synchronization in UltraLite applications

You synchronize an UltraLite database with a central consolidated database. Synchronization requires the MobiLink synchronization software included with SQL Anywhere.

This section provides a brief introduction to synchronization and describes some features of particular interest to users of UltraLite.NET.

For more information about synchronization, see “UltraLite Clients” [[MobiLink - Client Administration](#)].

You can also find a working example of synchronization in the CustDB sample application. For more information, see the *Samples\UltraLite.NET\CustDB* subdirectory of your SQL Anywhere 10 installation.

UltraLite.NET supports TCP/IP, HTTP, HTTPS, and TLS (transport-layer security) synchronization. Synchronization is initiated by the UltraLite application. In all cases, you use properties of the SyncParms object to control synchronization.

Separately licensed component required

ECC encryption and FIPS-certified encryption require a separate license. All strong encryption technologies are subject to export regulations.

See “Separately licensed components” [[SQL Anywhere 10 - Introduction](#)].

Initiating Synchronization in C# Application

The following code illustrates how to initiate synchronization in an application written in C#.

```
private void Sync()
{
    // Sync
    try
    {
        // This line would be used to synchronize a specific publication
        // called "high_priority"
        conn.SyncParms.PublicationMask = conn.Schema.GetPublicationSchema
        ("high_priority").Mask;

        // Set the synchronization parameters
        conn.SyncParms.Version = "Version1";
        conn.SyncParms.StreamParms = "";
        conn.SyncParms.Stream = ULStreamType.TCPIP;
        conn.SyncParms.UserName = "51";
        conn.Synchronize();
    }
    catch (System.Exception t)
    {
        MessageBox.Show("Exception: " + t.Message);
    }
}
```

Adding ActiveSync synchronization to your application

This section describes how to add ActiveSync synchronization to an UltraLite.NET application, and how to register your application for use with ActiveSync on your end users' computers.

ActiveSync synchronization can only be initiated by ActiveSync. ActiveSync initiates synchronization when the device is placed in the cradle or when the Synchronization command is selected from the ActiveSync window.

When ActiveSync initiates synchronization, the MobiLink provider for ActiveSync starts the UltraLite application, if it is not already running, and sends a message to it. Your application must implement a `ULActiveSyncListener` object to receive and process messages from the MobiLink provider. Your application must specify the listener object using the `SetActiveSyncListener` method, where **MyAppClassName** is a unique Windows class name for the application.

```
dbMgr.SetActiveSyncListener(  
    "MyAppClassName", listener );
```

For more information, including sample code, see [“ULActiveSyncListener interface” on page 53](#).

When UltraLite receives an ActiveSync message, it invokes the specified listener's `ActiveSyncInvoked` method on a different thread. To avoid multi-threading issues, your `ActiveSyncInvoked` method should post an event to the user interface.

If your application is multi-threaded, use a separate connection and use the **lock** keyword in C# or **SyncLock** keyword in Visual Basic .NET to access any objects shared with the rest of the application. The `ActiveSyncInvoked` method should specify a `ULStreamType.ACTIVE_SYNC` for its connection's `SyncParms.Stream` and then call `ULConnection.Synchronize`.

When registering your application, set the following parameter:

- ◆ **Class Name** The same class name the application used with the `Connection.SetActiveSyncListener` method.

CHAPTER 4

Tutorial: Build an UltraLite.NET Application

Contents

Introduction to UltraLite.NET development tutorial 36

Lesson 1: Create a Visual Studio project 37

Lesson 2: Create an UltraLite database 40

Lesson 3: Connect to the database 41

Lesson 4: Insert, update, and delete data 43

Lesson 5: Build and deploy application 48

Introduction to UltraLite.NET development tutorial

This tutorial guides you through the process of building an UltraLite.NET application using Microsoft Visual Studio. It uses the ADO.NET interface provided by the iAnywhere.Data.UltraLite namespace.

This tutorial contains code for a Visual Basic .NET application and a Visual C# application.

Competencies and experience

This tutorial assumes the following:

- ◆ You are familiar with the C# programming language or the Visual Basic .NET programming language.
- ◆ You have Microsoft Visual Studio installed on your computer and you are familiar with using Visual Studio. This tutorial was developed using Visual Studio .NET 2003 and may refer to Visual Studio actions or procedures that may be slightly different in other versions of Visual Studio.
- ◆ You know how to create an UltraLite database using UltraLite in Sybase Central.

See “Creating an UltraLite database from Sybase Central” [[UltraLite - Database Management and Reference](#)].

Goals

The goal for the tutorial is to gain competence and familiarity with the process of developing UltraLite.NET applications in the Visual Studio environment.

Installation note

If you install UltraLite software on a Windows computer that already has Visual Studio installed, the UltraLite installation process detects the presence of Visual Studio and performs the necessary integration steps. If you install Visual Studio after installing UltraLite, or install a new version of Visual Studio, the process to integrate UltraLite with Visual Studio must be performed manually as follows:

- ◆ Ensure Visual Studio is not running.
- ◆ For Visual Studio .NET 2003, run *installULNet.exe* from the folder named *install-dir\ultralite\UltraLite.NET\assembly\v1*.

For Visual Studio 2005, run *installULNet.exe* from the folder named *install-dir\ultralite\UltraLite.NET\assembly\v2*.

Lesson 1: Create a Visual Studio project

The following procedure creates and configures a new Visual Studio application. You can choose whether to use Visual Basic.NET or C# as your programming language.

♦ To create a Visual Studio project

1. Create a Visual Studio project.

- ♦ From the Visual Studio File menu, choose New ► Project to create a new project.

The New Project window appears.

- ♦ In the left pane, select either the Visual Basic Projects folder or the Visual C# Projects folder.

In the right pane, select a Smart Device Application and name your project **VBApp** or **CSApp**, depending on whether you are using Visual Basic or C# for the programming language.

- ♦ Enter a Location of *c:\tutorial\uldotnet* and click OK.

The Smart Device Application wizard appears.

- ♦ Choose Pocket PC as the target platform, and select Windows Application as the project type. Click OK.

A Design workspace appears, displaying a form named Form1.

2. Add references to your project.

- ♦ Add the iAnywhere.Data.UltraLite assembly and the associated resources to your project.

- a. From the Project menu, choose Add Reference.

The Add Reference window appears.

- b. Select **iAnywhere.Data.UltraLite (CE)** from the list of available references. Click Select to add it to the list of selected components.

If this reference does not appear in the list, click Browse and locate it in the *ultralite \UltraLite.NET\ce* subdirectory of your SQL Anywhere installation. Select *iAnywhere.Data.UltraLite.dll* and click Open.

- c. Select **iAnywhere.Data.UltraLite (CE) EN** from the list of available references. Click Select to add it to the list of selected components.

If this reference does not appear in the list, click Browse and locate it in the *ultralite \UltraLite.NET\ce\xx* subdirectory of your SQL Anywhere installation, where xx is a two-letter abbreviation for the language. Select *iAnywhere.Data.UltraLite.resources.dll* and click Open.

- d. Click OK to add the assembly and resources to your project.

- ♦ Link the UltraLite component to your project.

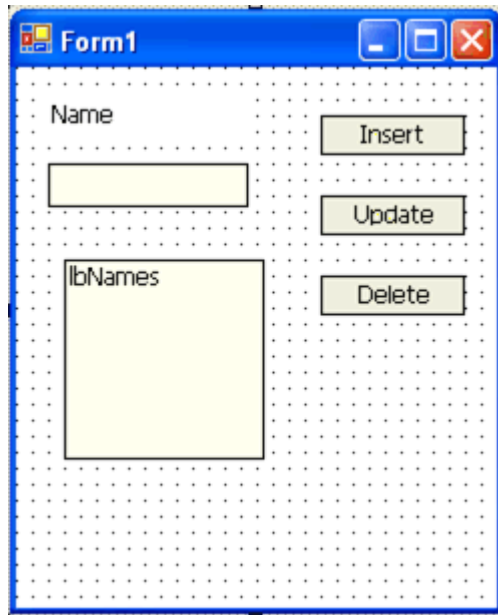
In this step, ensure that you add a link to the component, and that you do not open the component.

- a. From the Project menu, choose Add Existing Item and browse to the *ultralite\UltraLite.NET\ce* subdirectory of your SQL Anywhere installation.
 - b. Choose All Files from the Files of Type dropdown list so that DLL files appear.
 - c. Open the folder corresponding to the processor of the CE device you are using. For Pocket PC emulator in Visual Studio .NET 2003 open the x86 folder, but for Visual Studio 2005 open the ARM folder. Select *ulnet10.dll*; do not click Open.
 - d. Click the arrow on the Open button and select **Link File** to link it to your project.
- ♦ If you are using Visual Studio 2005, set the Build Action property of the dll to Content and set the Copy to Output Directory property to Copy if newer.
3. Create a form for your application.

If the Visual Studio toolbox panel is not currently displayed, choose View ► Toolbox from the main menu. Add the following visual components to the form by selecting the object from the toolbox and dragging it onto the form in the desired location.

Type	Name	Text
Button	btnInsert	Insert
Button	btnUpdate	Update
Button	btnDelete	Delete
TextBox	txtName	(no text)
ListBox	lbNames	(no text)
Label	laName	Name

Your form should look like the following figure:



4. Build and deploy your solution.

Building and deploying the solution confirms that you have configured your Visual Studio.NET project properly.

- a. From the Build menu, choose Build Solution. Confirm that the project builds successfully. If you are building a Visual Basic application, you can ignore the following warning that may appear:

Referenced assembly 'iAnywhere.Data.UltraLite.resources' is a localized satellite assembly

- b. From the Debug menu, choose Start.

This action deploys your application to the device or emulator, and starts it. The application is deployed to *\Program Files\VBApp* or *\Program Files\CSApp* depending on your project name.

The deployment may take some time.

- c. Confirm that the application deploys to the emulator or your target device.
- d. From the Debug menu, choose Stop Debugging to close the application.

Lesson 2: Create an UltraLite database

The following procedure creates an UltraLite database using Sybase Central.

See “Creating an UltraLite database from Sybase Central” [[UltraLite - Database Management and Reference](#)].

♦ To create a database

1. Start Sybase Central:

From the Start menu, choose Programs ► SQL Anywhere 10 ► Sybase Central.

2. Use the UltraLite plug-in in Sybase Central to create a database in the same directory as your application with the following characteristics:

♦ **Database file name** *VBApp.udb* or *CSApp.udb*, depending on your application.

♦ **Collation sequence** Use the default collation.

♦ **Case sensitivity** Leave this option cleared.

♦ **Table name** Type Names.

♦ **Columns** Create columns in the Names table with the following attributes:

Column Name	Data Type (Size)	Allow NULL?	Default value
ID	integer	No	global autoincrement
Name	varchar(30)	No	None

♦ **Primary key** Specify the ID column, ascending.

3. Link the initialized (empty) database file to your project so that the database file is deployed to the device along with the application code:

♦ From the Visual Studio menu, choose Project ► Add Existing Item.

♦ Browse to the directory where you created the database file and select the file *VBApp.udb* or *CSApp.udb* depending on your application. *Do not click Open.*

♦ Click the arrow in the Open button and click Link File.

♦ In the Solution Explorer frame, right click the database file name that has just been added to the project and choose Properties.

In the properties panel, set the Build Action property to Content; if you are using Visual Studio 2005, set the Copy to Output Directory property to Copy Always.

Lesson 3: Connect to the database

The following procedure adds a control to your UltraLite.NET application that establishes a connection to an UltraLite database.

This tutorial assumes that if you are designing a C# application, your files are in the directory *c:\tutorial\uldotnet\CSApp* and that if you are designing a Visual Basic application, your files are in the directory *c:\tutorial\uldotnet\VBApp*. If you created a directory with a different name, use that directory throughout the tutorial.

◆ To add an UltraLite connection to your application

1. Open the source code for the form.

Double-click the form to open the source file (*Form1.cs*).

2. Import the *iAnywhere.Data.UltraLite* namespace.

Add the following statement as the very first line of the file.

```
//Visual C#  
using iAnywhere.Data.UltraLite;  
  
'Visual Basic  
Imports iAnywhere.Data.UltraLite
```

3. Add global variables to the form declaration.

For a Visual C# application, add the following code after the code describing the form components and before the first method declaration.

```
//Visual C#  
private ULConnection Conn;  
private int[] ids;
```

For a Visual Basic project, add the following code after the declaration of the form components (FriendsWithEvents declarations) and before the *Form1* method declaration.

```
'Visual Basic  
Dim Conn As ULConnection  
Dim ids() As Integer
```

These variables are used as follows:

- ◆ **ULConnection** A Connection object is the root object for all actions executed on a connection to a database.

- ◆ **ids** The *ids* array is used to hold the ID column values returned after executing a query.

Although the Listbox control itself allows you access to sequential numbers, those numbers differ from the value of the ID column once a row has been deleted. For this reason, the ID column values must be stored separately.

4. Double-click a blank area of your form to create a *Form1_Load* method.

This method performs the following tasks:

- ◆ Opens a connection to the database using the connection parameters set in the `ulConnectionParms1` control.
- ◆ Calls the `RefreshListBox` method, which you will define later in this tutorial.
- ◆ If an error occurs, prints the error message. For SQL errors, the code also prints the error code.

See [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

For C#, add the following code to the `Form1_Load` method.

```
//Visual C#
try {
    String ConnString = "dbf=\\Program Files\\CSApp\\CSApp.udb";
    Conn = new ULConnection( ConnString );
    Conn.Open();
    Conn.DatabaseID = 1;
    RefreshListBox();
}
catch ( System.Exception t ) {
    MessageBox.Show( "Exception: " + t.Message);
}
```

For Visual Basic, add the following code to the method.

```
'Visual Basic
Try
    Dim ConnString as String = "dbf=\\Program Files\\VBApp\\VBApp.udb"
    Conn = New ULConnection( ConnString )
    Conn.Open()
    Conn.DatabaseID = 1
    RefreshListBox()
Catch
    MsgBox("Exception: " + err.Description)
End Try
```

5. Build the project.

From the Build menu choose Build Solution. At this stage, you may receive a single error reported; for example in C#: error CS0103: The name 'RefreshListBox' does not exist in the class or namespace 'CSApp.Form1' because RefreshListBox is not declared. The next lesson adds that function.

If you get other errors, you must correct them before proceeding. Check for common errors, such as case inconsistencies in C#. For example, **UltraLite** and **ULConnection** must match case exactly.

Lesson 4: Insert, update, and delete data

In this lesson you add code to your application to modify the data in your database. The following procedures use Dynamic SQL. The same techniques can be performed using the Table API.

See [“Accessing and manipulating data with the Table API” on page 22](#).

The following procedure creates a supporting method to maintain the listbox. This method is required for data manipulation methods created in the remaining procedures.

◆ To add code to maintain the listbox

1. Right-click the form and choose View Code.
2. Add a method of the Form1 class to update and populate the listbox. This method carries out the following tasks:
 - ◆ Clears the listbox.
 - ◆ Instantiates a ULCommand object and assigns it a SELECT query that returns data from the Names table in the database.
 - ◆ Executes the query, returning a result set as a ULDataReader.
 - ◆ Instantiates an integer array with length equal to the number of rows in the result set.
 - ◆ Populates the listbox with the names returned in the ULDataReader and populates the integer array with the ids returned in the ULDataReader.
 - ◆ Closes the ULDataReader.
 - ◆ If an error occurs, prints the error message. For SQL errors, the code also prints the error code.

See [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

For C#, add the following code to your application as a method of the Form1 class.

```
//Visual C#
private void RefreshListBox(){
    try{
        long NumRows;
        int i = 0;
        lbNames.Items.Clear();
        using( ULCommand cmd = Conn.CreateCommand() ){
            cmd.CommandText = "SELECT ID, Name FROM Names";
            using( ULDataReader dr = cmd.ExecuteReader() ){
                dr.MoveBeforeFirst();
                NumRows = dr.RowCount;
                ids = new int[ NumRows ];
                while (dr.MoveNext())
                {
                    lbNames.Items.Add(
                        dr.GetString(1));
                    ids[ i ] = dr.GetInt32(0);
                    i++;
                }
            }
        }
    }
}
```

```
        }
        txtName.Text = " ";
    }
}
catch( Exception err ){
    MessageBox.Show(
        "Exception in RefreshListBox: " + err.Message );
}
}
```

For Visual Basic, add the following code to your application as a method of the Form1 class.

```
'Visual Basic
Private Sub RefreshListBox()
    Try
        Dim cmd As ULCommand = Conn.CreateCommand()
        Dim i As Integer = 0
        lbNames.Items.Clear()
        cmd.CommandText = "SELECT ID, Name FROM Names"
        Dim dr As ULDataReader = cmd.ExecuteReader()
        ReDim ids(dr.RowCount)
        While (dr.MoveNext)
            lbNames.Items.Add(dr.GetString(1))
            ids(i) = dr.GetInt32(0)
            i = i + 1
        End While
        dr.Close()
        txtName.Text = " "
    Catch ex As Exception
        MsgBox(ex.ToString)
    End Try
End Sub
```

3. Build the project.

Building the project should result in no errors.

◆ To implement INSERT, UPDATE, and DELETE

1. Double-click the Insert button to create a btnInsert_Click method. This method carries out the following tasks:

- ◆ Instantiates a ULCommand object and assigns it an INSERT statement that inserts the value in the text box into the database.
- ◆ Executes the statement.
- ◆ Disposes of the ULCommand object.
- ◆ Refreshes the listbox.
- ◆ If an error occurs, prints the error message. For SQL errors, the code also prints the error code.

See [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

For C#, add the following code to the method.

```
//Visual C#
try {
    long RowsInserted;
```

```

        using( ULCommand cmd = Conn.CreateCommand() ) {
            cmd.CommandText =
                "INSERT INTO Names(name) VALUES (?)";
            cmd.Parameters.Add("", txtName.Text);
            RowsInserted = cmd.ExecuteNonQuery();
        }
        RefreshListBox();
    }
    catch( Exception err ) {
        MessageBox.Show("Exception: " + err.Message );
    }
}

```

For Visual Basic, add the following code to the method.

```

'Visual Basic
Try
    Dim RowsInserted As Long
    Dim cmd As ULCommand = Conn.CreateCommand()
    cmd.CommandText = "INSERT INTO Names(name) VALUES (?)"
    cmd.Parameters.Add("", txtName.Text)
    RowsInserted = cmd.ExecuteNonQuery()
    cmd.Dispose()
    RefreshListBox()
Catch
    MsgBox("Exception: " + Err.Description)
End Try

```

2. Double-click the Update button to create a btnUpdate_Click method. This method carries out the following tasks:
 - ◆ Instantiates a ULCommand object and assigns it an UPDATE statement that inserts the value in the text box into the database based on the associated ID.
 - ◆ Executes the statement.
 - ◆ Disposes of the ULCommand object.
 - ◆ Refreshes the listbox.
 - ◆ If an error occurs, prints the error message. For SQL errors, the code also prints the error code.

See [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

For C#, add the following code to the method.

```

//Visual C#
try {
    long RowsUpdated;
    int updateID = ids[ lbNames.SelectedIndex ];
    using( ULCommand cmd = Conn.CreateCommand() ){
        cmd.CommandText =
            "UPDATE Names SET name = ? WHERE id = ?" ;
        cmd.Parameters.Add("", txtName.Text );
        cmd.Parameters.Add("", updateID);
        RowsUpdated = cmd.ExecuteNonQuery();
    }
    RefreshListBox();
}
catch( Exception err ) {
    MessageBox.Show(

```

```
        "Exception: " + err.Message);  
    }
```

For Visual Basic, add the following code to the method.

```
'Visual Basic  
Try  
    Dim RowsUpdated As Long  
    Dim updateID As Integer = ids(lbNames.SelectedIndex)  
    Dim cmd As ULCommand = Conn.CreateCommand()  
    cmd.CommandText = "UPDATE Names SET name = ? WHERE id = ?"  
    cmd.Parameters.Add("", txtName.Text)  
    cmd.Parameters.Add("", updateID)  
    RowsUpdated = cmd.ExecuteNonQuery()  
    cmd.Dispose()  
    RefreshListBox()  
Catch  
    MsgBox("Exception: " + Err.Description)  
End Try
```

3. Double-click the Delete button to create a btnDelete_Click method. Add code to carry out the following tasks:

- ◆ Instantiates a ULCommand object and assigns it a DELETE statement. The DELETE statement deletes the selected row from the database, based on the associated ID from the integer array ids.
- ◆ Executes the statement.
- ◆ Disposes of the ULCommand object.
- ◆ Refreshes the listbox.
- ◆ If an error occurs, displays the error message. For SQL errors, the code also displays the error code.

See [SQL Anywhere 10 - Error Messages \[SQL Anywhere 10 - Error Messages\]](#).

For C#, add the following code to the method.

```
//Visual C#  
try{  
    long RowsDeleted;  
    int deleteID = ids[lbNames.SelectedIndex];  
    using( ULCommand cmd = Conn.CreateCommand() ){  
        cmd.CommandText =  
            "DELETE From Names WHERE id = ?" ;  
        cmd.Parameters.Add("", deleteID);  
        RowsDeleted = cmd.ExecuteNonQuery ();  
    }  
    RefreshListBox();  
}  
catch( Exception err ) {  
    MessageBox.Show("Exception: " + err.Message );  
}
```

For Visual Basic, add the following code to the method.

```
'Visual Basic  
Try  
    Dim RowsDeleted As Long  
    Dim deleteID As Integer = ids(lbNames.SelectedIndex)
```



```
Dim cmd As ULCommand = Conn.CreateCommand()  
cmd.CommandText = "DELETE From Names WHERE id = ?"  
cmd.Parameters.Add("", deleteID)  
RowsDeleted = cmd.ExecuteNonQuery()  
cmd.Dispose()  
RefreshListBox()  
Catch  
    MsgBox("Exception: " + Err.Description)  
End Try
```

4. Build your application to confirm that it compiles properly.

Lesson 5: Build and deploy application

In the following procedure, you build your application and deploy it to a remote device.

♦ To deploy your application

1. Build the solution.

Ensure that your application builds without errors.

2. Choose the deployment target.

The deployment target must match the version of *ulnet10.dll* that you included in your application.

3. Choose Debug ► Start.

This builds an executable file containing your application and deploys it to the Pocket PC emulator. The process may take some time, especially if it must deploy the .NET Compact Framework before running the application.

Deployment troubleshooting checklist

If errors are reported, you may want to check that your deployment was completed successfully using the following checklist:

- ♦ Confirm that the application is deployed into *\Program Files\appname*, where *appname* is the name you gave your application in Lesson 1 (CSApp or VBApp).
- ♦ Confirm that the path to the database file in your application code is correct. See [“Lesson 3: Connect to the database” on page 41](#).
- ♦ Confirm that you chose Link File when adding the database file to the project and you set the Build Action to Content. If using Visual Studio 2005, confirm that you set Copy to Output Directory to Copy Always or Copy if Newer. If you did not set these options correctly, the files will not be deployed to the device.
- ♦ Ensure that you added a reference to the correct version of *ulnet10.dll* for your target platform, or ran the Windows CE installer. For versions of Windows Mobile earlier than Windows Mobile 5.0, if you switch between the emulator and a real device, you must change the version of the library that you use. See [“Lesson 1: Create a Visual Studio project” on page 37](#).
- ♦ You may want to exit the emulator without saving the emulator state. Redeploying the application copies all required files to the emulator, and ensures there are no version problems.

♦ To test your application

1. Insert data into the database.

Enter a name in the text box and click Insert. The name should now appear in the listbox.

2. Update data in the database.

Select a name from the listbox. Enter a new name in the text box. Click Update. The new name should now appear in place of the old name in the listbox.

3. Delete data from the database.

Select a name from the listbox. Click Delete. The name should no longer appear in the listbox.

This completes the tutorial.

CHAPTER 5

UltraLite .NET 1.0 API Reference

Contents

ULActiveSyncListener interface	53
ULAuthStatusCode enumeration	56
ULCommand class	57
ULCommandBuilder class	82
ULConnection class	90
ULConnectionParms class	122
ULConnectionParms.UnusedEventHandler delegate	133
ULCreateParms class	134
ULCursorSchema class	144
ULDataAdapter class	153
ULDatabaseManager class	178
ULDatabaseSchema class	185
ULDataReader class	195
ULDateOrder enumeration	237
ULDbType enumeration	238
ULException class	242
ULFileTransfer class	246
ULFileTransferProgressData class	260
ULFileTransferProgressListener interface	263
ULIndexSchema class	265
ULInfoMessageEventArgs class	273
ULInfoMessageEventHandler delegate	276
ULParameter class	277
ULParameterCollection class	294
ULPublicationSchema class	312
ULResultSet class	316
ULResultSetSchema class	342
ULRowUpdatedEventArgs class	344
ULRowUpdatedEventHandler delegate	348

ULRowUpdatingEventArgs class	349
ULRowUpdatingEventHandler delegate	352
ULRuntimeType enumeration	353
ULServerSyncListener interface	354
ULSQLCode enumeration	357
ULStreamErrorCode enumeration	366
ULStreamErrorContext enumeration	381
ULStreamErrorID enumeration	382
ULStreamType enumeration	384
ULSyncParms class	385
ULSyncProgressData class	399
ULSyncProgressListener interface	409
ULSyncProgressState enumeration	411
ULSyncResult class	413
ULTable class	419
ULTableSchema class	441
ULTransaction class	456

ULActiveSyncListener interface

UL Ext.: The listener interface for receiving ActiveSync events.

Syntax

Visual Basic

Public Interface **ULActiveSyncListener**

C#

public interface **ULActiveSyncListener**

ULActiveSyncListener members

Public methods

Member name	Description
ActiveSyncInvoked method	Invoked when the MobiLink provider for ActiveSync calls the application to perform synchronization.

ActiveSyncInvoked method

Invoked when the MobiLink provider for ActiveSync calls the application to perform synchronization.

Syntax

Visual Basic

```
Public Sub ActiveSyncInvoked( _  
    ByVal launchedByProvider As Boolean _  
)
```

C#

```
public void ActiveSyncInvoked(  
    bool launchedByProvider  
);
```

Parameters

- ◆ **launchedByProvider** True if the application was launched by the MobiLink provider to perform ActiveSync synchronization. The application must then shut itself down after it has finished synchronizing. False if the application was already running when called by the MobiLink provider for ActiveSync.

Remarks

This method is invoked by a separate thread. To avoid multi-threading issues, it should post an event to the UI. If you are using multi-threading, it is recommended that you use a separate connection and use the lock keyword to access any objects shared with the rest of the application.

Once synchronization has completed, applications should call `ULDatabaseManager.SignalSyncIsComplete()` to signal the MobiLink provider for ActiveSync.

Example

The following code fragments demonstrate how to receive an ActiveSync request and perform a synchronization in the UI thread.

```
' Visual Basic
Imports iAnywhere.Data.UltraLite

Public Class MainWindow
    Inherits System.Windows.Forms.Form
    Implements ULAActiveSyncListener
    Private conn As ULConnection

    Public Sub New(ByVal args() As String)

        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        ULConnection.DatabaseManager.SetActiveSyncListener( _
            "myCompany.myapp", Me _
        )
        'Create Connection
        ...
    End Sub

    Protected Overrides Sub OnClosing( _
        ByVal e As System.ComponentModel.CancelEventArgs _
    )
        ULConnection.DatabaseManager.SetActiveSyncListener( _
            Nothing, Nothing _
        )
        MyBase.OnClosing(e)
    End Sub

    Public Sub ActiveSyncInvoked( _
        ByVal launchedByProvider As Boolean _
    ) Implements ULAActiveSyncListener.ActiveSyncInvoked
        Me.Invoke(New EventHandler(AddressOf Me.ActiveSyncAction))
    End Sub

    Public Sub ActiveSyncAction( _
        ByVal sender As Object, ByVal e As EventArgs _
    )
        ' Do active sync
        conn.Synchronize()
        ULConnection.DatabaseManager.SignalSyncIsComplete()
    End Sub
End Class

// C#
using iAnywhere.Data.UltraLite;
```



```
public class Form1 : System.Windows.Forms.Form, ULActiveSyncListener
{
    private System.Windows.Forms.MainMenu mainMenu1;
    private ULConnection conn;

    public Form1()
    {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after
        // InitializeComponent call
        //
        ULConnection.DatabaseManager.SetActiveSyncListener(
            "myCompany.myapp", this
        );
        // Create connection
        ...
    }

    protected override void Dispose( bool disposing )
    {
        base.Dispose( disposing );
    }

    protected override void OnClosing(
        System.ComponentModel.CancelEventArgs e
    )
    {
        ULConnection.DatabaseManager.SetActiveSyncListener(
            null, null
        );
        base.OnClosing(e);
    }

    public void ActiveSyncInvoked(bool launchedByProvider)
    {
        this.Invoke( new EventHandler( ActiveSyncHandler ) );
    }

    internal void ActiveSyncHandler(object sender, EventArgs e)
    {
        conn.Synchronize();
        ULConnection.DatabaseManager.SignalSyncIsComplete();
    }
}
```

See also

- ◆ [“ULActiveSyncListener interface” on page 53](#)
- ◆ [“ULActiveSyncListener members” on page 53](#)
- ◆ [“SignalSyncIsComplete method” on page 184](#)

ULAuthStatusCode enumeration

UL Ext.: Enumerates the status codes that may be reported during MobiLink user authentication.

Syntax

Visual Basic

Public Enum **ULAuthStatusCode**
Inherits Short

C#

```
public enum ULAuthStatusCode :  
short
```

Member name

Member name	Description
EXPIRED	User ID or password has expired - authorization failed (EXPIRED = 3).
IN_USE	User ID is already in use - authorization failed (IN_USE = 5).
INVALID	Bad user ID or password - authorization failed (INVALID = 4).
UNKNOWN	Authorization status is unknown, possibly because the connection has not yet performed a synchronization (UNKNOWN = 0).
VALID	User ID and password were valid at time of synchronization (VALID = 1).
VALID_BUT_EXPIRES_SOON	User ID and password were valid at time of synchronization, but will expire soon (VALID_BUT_EXPIRES_SOON = 2).

See also

- ◆ [“AuthStatus property” on page 414](#)

ULCommand class

Represents a pre-compiled SQL statement or query, with or without IN parameters. This object can be used to execute a statement or query efficiently multiple times.

Syntax

Visual Basic

NotInheritable Public Class **ULCommand**
Inherits Component

C#

public sealed class **ULCommand** :
Component

Remarks

ULCommand objects can be created directly, or with the ULConnection.CreateCommand method. This method ensures that the command has the correct transaction for executing statements on the given connection.

The ULCommand.Transaction method must be reset after the current transaction is committed or rolled back.

ULCommand features the following methods for executing commands on an UltraLite.NET database:

Method	Description
ULCommand.ExecuteNonQuery	Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.
ULCommand.ExecuteReader()	Executes a SQL SELECT statement and returns the result set in a ULDataReader. Use this method for creating read-only result sets.
ULCommand.ExecuteResultSet() ()	UL Ext.: Executes a SQL SELECT statement and returns the result set in a ULResultSet. Use this method for creating mutable result sets.
ULCommand.ExecuteScalar	Executes a SQL SELECT statement and returns a single value.
ULCommand.ExecuteTable()	UL Ext.: Retrieves a database table in a ULTable for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and the ULCommand.IndexName can be used to specify a table sorting order. The ULCommand.CommandType must be System.Data.CommandType.TableDirect.

You can reset most properties, including the ULCommand.CommandText, and reuse the ULCommand object.

For resource management reasons, it is recommended that you explicitly dispose of commands when you are done with them. In C#, you may use a using statement to automatically call the System.ComponentModel.Component.Dispose() method or explicitly call the

System.ComponentModel.Component.Dispose() method. In Visual Basic, you always explicitly call the System.ComponentModel.Component.Dispose() method.

Implements: System.Data.IDbCommand, System.IDisposable

See also

- ◆ [“ULCommand members” on page 58](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“ExecuteNonQuery method” on page 71](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“ExecuteResultSet method” on page 74](#)
- ◆ [“ExecuteScalar method” on page 77](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [System.ComponentModel.Component.Dispose](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [“IndexName property” on page 66](#)
- ◆ [IDbCommand](#)
- ◆ [IDisposable](#)
- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULDataReader class” on page 195](#)

ULCommand members

Public constructors

Member name	Description
ULCommand constructor	Initializes a ULCommand object.
ULCommand constructor	Initializes a ULCommand object with the specified command text.
ULCommand constructor	Initializes a ULCommand object with the specified command text and connection.
ULCommand constructor	Initializes a ULCommand object with the specified command text, connection, and transaction.

Public properties

Member name	Description
CommandText property	Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.

Member name	Description
CommandTimeout property	This feature is not supported by UltraLite.NET.
CommandType property	Specifies the type of command to be executed.
Connection property	The connection object on which to execute the ULCommand object.
DesignTimeVisible property	Indicates if the ULCommand should be visible in a customized Windows Form Designer control.
IndexName property	UL Ext.: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType is System.Data.CommandType.TableDirect.
Parameters property	Specifies the parameters for the current statement.
Plan property	UL Ext.: Returns the access plan UltraLite.NET uses to execute a query. This property is intended primarily for use during development.
Transaction property	Specifies the ULTransaction in which the ULCommand executes.
UpdatedRowSource property	Specifies how command results are applied to the DataRow when used by the Update method of the ULDataAdapter.

Public methods

Member name	Description
Cancel method	This method is not supported in UltraLite.NET.
CreateParameter method	Provides a ULParameter object for supplying parameters to ULCommand objects.
ExecuteNonQuery method	Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.
ExecuteReader method	Executes a SQL SELECT statement and returns the result set.
ExecuteReader method	Executes a SQL SELECT statement with the specified command behavior and returns the result set.
ExecuteResultSet method	UL Ext.: Executes a SQL SELECT statement and returns the result set as a ULResultSet.
ExecuteResultSet method	UL Ext.: Executes a SQL SELECT statement with the specified command behavior and returns the result set as a ULResultSet.
ExecuteScalar method	Executes a SQL SELECT statement and returns a single value.
ExecuteTable method	UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and ULCommand.IndexName can be used to specify a table sorting order.

Member name	Description
ExecuteTable method	UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The <code>ULCommand.CommandText</code> is interpreted as the name of the table and <code>ULCommand.IndexName</code> can be used to specify a table sorting order.
Prepare method	Pre-compiles and stores the SQL statement of this command.

ULCommand constructor

Initializes a `ULCommand` object.

Syntax

Visual Basic

Overloads Public Sub **New()**

C#

public **ULCommand()**;

Remarks

The `ULCommand` object needs to have the `ULCommand.CommandText`, `ULCommand.Connection`, and `ULCommand.Transaction` properties set before a statement can be executed.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [“ULCommand constructor” on page 60](#)
- ◆ [“ULCommand constructor” on page 61](#)
- ◆ [“ULCommand constructor” on page 62](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)

ULCommand constructor

Initializes a `ULCommand` object with the specified command text.

Syntax

Visual Basic

Overloads Public Sub **New**(
 ByVal *cmdText* As String
)

C#

```
public ULCommand(  
    string cmdText  
);
```

Parameters

- ◆ **cmdText** The text of the SQL statement or name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.

Remarks

The ULCommand object needs to have the ULCommand.Connection and ULCommand.Transaction set before a statement can be executed.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [“ULCommand constructor” on page 60](#)
- ◆ [“ULCommand constructor” on page 61](#)
- ◆ [“ULCommand constructor” on page 62](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)

ULCommand constructor

Initializes a ULCommand object with the specified command text and connection.

Syntax**Visual Basic**

```
Overloads Public Sub New( _  
    ByVal cmdText As String, _  
    ByVal connection As ULConnection _  
)
```

C#

```
public ULCommand(  
    string cmdText,  
    ULConnection connection  
);
```

Parameters

- ♦ **cmdText** The text of the SQL statement or name of the table when the `ULCommand.CommandType` is `System.Data.CommandType.TableDirect`. For parameterized statements, use a question mark (?) placeholder to pass parameters.
- ♦ **connection** The `ULConnection` object representing the current connection.

Remarks

The `ULCommand` object may need to have the `ULCommand.Transaction` set before a statement can be executed.

See also

- ♦ [“ULCommand class” on page 57](#)
- ♦ [“ULCommand members” on page 58](#)
- ♦ [“CreateCommand method” on page 108](#)
- ♦ [“ULCommand constructor” on page 60](#)
- ♦ [“ULCommand constructor” on page 60](#)
- ♦ [“ULCommand constructor” on page 62](#)
- ♦ [“Transaction property” on page 68](#)
- ♦ [“CommandType property” on page 64](#)
- ♦ [CommandType.TableDirect](#)
- ♦ [“ULConnection class” on page 90](#)

ULCommand constructor

Initializes a `ULCommand` object with the specified command text, connection, and transaction.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal cmdText As String, _  
    ByVal connection As ULConnection, _  
    ByVal transaction As ULTransaction _  
)
```

C#

```
public ULCommand(  
    string cmdText,  
    ULConnection connection,  
    ULTransaction transaction  
);
```

Parameters

- ♦ **cmdText** The text of the SQL statement or name of the table when the `ULCommand.CommandType` is `System.Data.CommandType.TableDirect`. For parameterized statements, use a question mark (?) placeholder to pass parameters.

- ◆ **connection** The ULConnection object representing the current connection.
- ◆ **transaction** The ULTransaction in which the ULCommand executes.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [“ULCommand constructor” on page 60](#)
- ◆ [“ULCommand constructor” on page 60](#)
- ◆ [“ULCommand constructor” on page 61](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULTransaction class” on page 456](#)

CommandText property

Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.

Syntax

Visual Basic

```
NotOverridable Public Property CommandText As String _  
    Implements IDbCommand.CommandText
```

C#

```
public string CommandText {get;set;}
```

Property value

A string specifying the text of the SQL statement or the name of the table. The default is an empty string (invalid command).

Remarks

It is recommended that SELECT statements used to create read-only result sets (ULCommand.ExecuteReader() or ULCommand.ExecuteScalar) should end with " FOR READ ONLY". For some statements that use temporary tables, there may be a significant performance improvement.

Implements

[IDbCommand.CommandText](#)

Example

The following example demonstrates the use of the parameterized placeholder:

```
' Visual Basic  
myCmd.CommandText = "SELECT * FROM Customers WHERE CustomerID = ?"
```

```
// C#  
myCmd.CommandText = "SELECT * FROM Customers WHERE CustomerID = ?";
```

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteNonQuery method” on page 71](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“ExecuteResultSet method” on page 74](#)
- ◆ [“ExecuteScalar method” on page 77](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)

CommandTimeout property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

NotOverridable Public Property **CommandTimeout** As Integer _
Implements IDbCommand.CommandTimeout

C#

public int **CommandTimeout** {get;set;}

Property value

The value is always zero.

Exceptions

- ◆ [ULException class](#) - Setting the value is not supported in UltraLite.NET.

Implements

[IDbCommand.CommandTimeout](#)

CommandType property

Specifies the type of command to be executed.

Syntax

Visual Basic

NotOverridable Public Property **CommandType** As CommandType _
Implements IDbCommand.CommandType

C#

```
public CommandType CommandType {get;set;}
```

Property value

One of the System.Data.CommandType values. The default is System.Data.CommandType.Text.

Remarks

Supported command types are as follows:

- ◆ System.Data.CommandType.TableDirect - **UL Ext.:** When you specify this CommandType, the ULCommand.CommandText must be the name of a database table. You can also specify the index used to open (sort) the table with ULCommand.IndexName. Use ULCommand.ExecuteTable() or ULCommand.ExecuteReader() to access the table.
- ◆ System.Data.CommandType.Text - When you specify this CommandType, the ULCommand.CommandText must be a SQL statement or query. Use ULCommand.ExecuteNonQuery to execute a non-query SQL statement and use either ULCommand.ExecuteReader() or ULCommand.ExecuteScalar to execute a query.

Exceptions

- ◆ [ArgumentException](#) - CommandType.StoredProcedure is not supported in UltraLite.NET.

Implements

[IDbCommand.CommandType](#)

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [CommandType](#)
- ◆ [CommandType.Text](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“IndexName property” on page 66](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [CommandType.Text](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ExecuteNonQuery method” on page 71](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“ExecuteScalar method” on page 77](#)

Connection property

The connection object on which to execute the ULCommand object.

Syntax**Visual Basic**

Public Property **Connection** As ULConnection

C#

public ULConnection **Connection** {get;set;}

Property value

The ULConnection object on which to execute the command.

Remarks

ULCommand objects must have an open connection before they can be executed.

The default is a null reference (Nothing in Visual Basic).

This is the strongly-typed version of System.Data.IDbCommand.Connection.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ULConnection class” on page 90](#)
- ◆ [IDbCommand.Connection](#)

DesignTimeVisible property

Indicates if the ULCommand should be visible in a customized Windows Form Designer control.

Syntax**Visual Basic**

Public Property **DesignTimeVisible** As Boolean

C#

public bool **DesignTimeVisible** {get;set;}

Property value

True if this ULCommand instance should be visible, false if this instance should not be visible. The default is false.

IndexName property

UL Ext.: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType is System.Data.CommandType.TableDirect.

Syntax

Visual Basic

Public Property **IndexName** As String

C#

public string **IndexName** {get;set;}

Property value

A string specifying the name of the index. The default is a null reference (Nothing in Visual Basic), meaning the table is opened with its primary key.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)

Parameters property

Specifies the parameters for the current statement.

Syntax

Visual Basic

Public Readonly Property **Parameters** As ULParameterCollection

C#

public ULParameterCollection **Parameters** {get;}

Property value

A ULParameterCollection holding the parameters of the SQL statement. The default value is the empty collection.

Remarks

Use question marks in ULCommand.CommandText to indicate parameters. The parameters in the collection are specified in the same order as the question mark placeholders. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in this collection.

This is the strongly-typed version of System.Data.IDbCommand.Parameters.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [IDbCommand.Connection](#)
- ◆ [“CommandText property” on page 63](#)

Plan property

UL Ext.: Returns the access plan UltraLite.NET uses to execute a query. This property is intended primarily for use during development.

Syntax

Visual Basic

Public Readonly Property **Plan** As String

C#

public string **Plan** {get;}

Property value

A string containing the text-based description of the query execution plan.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Transaction property

Specifies the ULTransaction in which the ULCommand executes.

Syntax

Visual Basic

Public Property **Transaction** As ULTransaction

C#

public ULTransaction **Transaction** {get;set;}

Property value

The ULTransaction in which the ULCommand executes. This should be the current transaction of the connection specified by the ULCommand.Connection. The default is a null reference (Nothing in Visual Basic).

Remarks

If a command is reused after a transaction has been committed or rolled back, this property needs to be reset.

This is the strongly-typed version of `System.Data.IDbCommand.Transaction`.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“BeginTransaction method” on page 102](#)
- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [IDbCommand.Transaction](#)

UpdatedRowSource property

Specifies how command results are applied to the `DataRow` when used by the `Update` method of the `ULDataAdapter`.

Syntax

Visual Basic

NotOverridable Public Property **UpdatedRowSource** As UpdateRowSource _
Implements IDbCommand.UpdatedRowSource

C#

```
public UpdateRowSource UpdatedRowSource {get;set;}
```

Property value

One of the `System.Data.UpdateRowSource` values. The default value is `System.Data.UpdateRowSource.Both`.

Implements

[IDbCommand.UpdatedRowSource](#)

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [UpdateRowSource](#)
- ◆ [UpdateRowSource.Both](#)

Cancel method

This method is not supported in UltraLite.NET.

Syntax**Visual Basic**

```
NotOverridable Public Sub Cancel() _  
    Implements IDbCommand.Cancel
```

C#

```
public void Cancel();
```

Remarks

This method does nothing. UltraLite.NET commands cannot be interrupted while they are executing.

Implements

[IDbCommand.Cancel](#)

CreateParameter method

Provides a ULParameter object for supplying parameters to ULCommand objects.

Syntax**Visual Basic**

```
Public Function CreateParameter() As ULParameter
```

C#

```
public ULParameter CreateParameter();
```

Return value

A new parameter, as a ULParameter object.

Remarks

Some SQL statements can take parameters, indicated in the text of a statement by a question mark (?). The CreateParameter method provides a ULParameter object. You can set properties on the ULParameter to specify the value for the parameter.

This is the strongly-typed version of System.Data.IDbCommand.CreateParameter.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [IDbCommand.CreateParameter](#)

ExecuteNonQuery method

Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.

Syntax

Visual Basic

```
NotOverridable Public Function ExecuteNonQuery() As Integer _  
    Implements IDbCommand.ExecuteNonQuery
```

C#

```
public int ExecuteNonQuery();
```

Return value

The number of rows affected.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and ULCommand.Parameters as needed.

For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, and for rollbacks, the return value is -1.

The ULCommand.CommandType cannot be System.Data.CommandType.TableDirect.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

Implements

[IDbCommand.ExecuteNonQuery](#)

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandText property” on page 63](#)

ExecuteReader method

Executes a SQL SELECT statement and returns the result set.

Syntax

Visual Basic

Overloads Public Function **ExecuteReader()** As ULDataReader

C#

public ULDataReader **ExecuteReader()**;

Return value

The result set as a ULDataReader object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULDataReader object is a read-only result set. For editable result sets, use ULCommand.ExecuteResultSet(), ULCommand.ExecuteTable(), or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable() and returns a ULTable downcast as a ULDataReader.

It is recommended that SELECT statements used with this method to create read-only result sets end with FOR READ ONLY. For some statements that use temporary tables, there may be a significant performance improvement.

This is the strongly-typed version of System.Data.IDbCommand.ExecuteReader().

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteReader method” on page 73](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ExecuteResultSet method” on page 74](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable method” on page 78](#)

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [IDbCommand.ExecuteReader](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)

ExecuteReader method

Executes a SQL SELECT statement with the specified command behavior and returns the result set.

Syntax

Visual Basic

```
Overloads Public Function ExecuteReader( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULDataReader
```

C#

```
public ULDataReader ExecuteReader(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

The result set as a ULDataReader object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULDataReader object is a read-only result set. For editable result sets, use ULCommand.ExecuteResultSet(CommandBehavior), ULCommand.ExecuteTable(CommandBehavior), or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable(CommandBehavior) and returns a ULTable downcast as a ULDataReader.

It is recommended that SELECT statements used with this method to create read-only result sets end with FOR READ ONLY. For some statements that use temporary tables, there may be a significant performance improvement.

This is the strongly-typed version of System.Data.IDbCommand.ExecuteReader (System.Data.CommandBehavior).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the `ULCommand.Connection` is missing or closed, the `ULCommand.Transaction` value does not match the current transaction state of the connection, or the `ULCommand.CommandText` is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ExecuteResultSet method” on page 75](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [IDbCommand.ExecuteReader](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandText property” on page 63](#)

ExecuteResultSet method

UL Ext.: Executes a SQL SELECT statement and returns the result set as a `ULResultSet`.

Syntax

Visual Basic

Overloads Public Function **ExecuteResultSet()** As `ULResultSet`

C#

public `ULResultSet` **ExecuteResultSet();**

Return value

The result set as a `ULResultSet` object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULResultSet object is an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use ULCommand.ExecuteTable() or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable() and returns a ULTable downcast as a ULResultSet.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteResultSet method” on page 75](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULTable class” on page 419](#)

ExecuteResultSet method

UL Ext.: Executes a SQL SELECT statement with the specified command behavior and returns the result set as a ULResultSet.

Syntax

Visual Basic

```
Overloads Public Function ExecuteResultSet( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULResultSet
```

C#

```
public ULResultSet ExecuteResultSet(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

The result set as a ULResultSet object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULResultSet object is an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use ULCommand.ExecuteTable (CommandBehavior) or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable(CommandBehavior) and returns a ULTable downcast as a ULResultSet.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteReader method” on page 72](#)
- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)

ExecuteScalar method

Executes a SQL SELECT statement and returns a single value.

Syntax

Visual Basic

NotOverridable Public Function **ExecuteScalar()** As Object _
Implements IDbCommand.ExecuteScalar

C#

public object **ExecuteScalar()**;

Return value

The first column of the first row in the result set, or a null reference (Nothing in Visual Basic) if the result set is empty.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required.

If this method is called on a query that returns multiple rows and columns, only the first column of the first row is returned.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteScalar performs an ULCommand.ExecuteTable() and returns the first column of the first row.

It is recommended that SELECT statements used with this method to create read-only result sets end with FOR READ ONLY. For some statements that use temporary tables, there may be a significant performance improvement.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

Implements

[IDbCommand.ExecuteScalar](#)

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“IndexName property” on page 66](#)
- ◆ [CommandBehavior](#)

- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [CommandType.TableDirect](#)

ExecuteTable method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and ULCommand.IndexName can be used to specify a table sorting order.

Syntax

Visual Basic

Overloads Public Function **ExecuteTable()** As ULTable

C#

public ULTable **ExecuteTable();**

Return value

The table as a ULTable object.

Remarks

The ULCommand.CommandType must be set to System.Data.CommandType.TableDirect.

If the ULCommand.IndexName is a null reference (Nothing in Visual Basic), the primary key is used to open the table. Otherwise, the table is opened using the ULCommand.IndexName value as the name of the index by which to sort.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the ULCommand.Connection is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“IndexName property” on page 66](#)
- ◆ [CommandBehavior](#)

- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULTable class” on page 419](#)

ExecuteTable method

UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The `ULCommand.CommandText` is interpreted as the name of the table and `ULCommand.IndexName` can be used to specify a table sorting order.

Syntax

Visual Basic

```
Overloads Public Function ExecuteTable( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of `System.Data.CommandBehavior` flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the `System.Data.CommandBehavior.Default`, `System.Data.CommandBehavior.CloseConnection`, and `System.Data.CommandBehavior.SchemaOnly` flags.

Return value

The table as a `ULTable` object.

Remarks

The `ULCommand.CommandType` must be set to `System.Data.CommandType.TableDirect`.

If the `ULCommand.IndexName` is a null reference (Nothing in Visual Basic), the primary key is used to open the table. Otherwise, the table is opened using the `ULCommand.IndexName` value as the name of the index by which to sort.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the `ULCommand.Connection` is missing or closed, the `ULCommand.Transaction` value does not match the current transaction state of the connection, or the `ULCommand.CommandText` is invalid.

See also

- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULCommand members” on page 58](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“Connection property” on page 65](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“IndexName property” on page 66](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 64](#)
- ◆ [CommandType.TableDirect](#)

Prepare method

Pre-compiles and stores the SQL statement of this command.

Syntax**Visual Basic**

```
NotOverridable Public Sub Prepare() _  
    Implements IDbCommand.Prepare
```

C#

```
public void Prepare();
```

Remarks

Pre-compiling statements allows for the efficient re-use of statements when just the parameter values are changed. Changing any other property on this command unprepares the statement.

UltraLite.NET does not require you to explicitly prepare statements as all unprepared commands are prepared on calls to the various Execute methods.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The command is in an invalid state. Either the `ULCommand.Connection` is missing or closed, the `ULCommand.Transaction` value does not match the current transaction state of the connection, or the `ULCommand.CommandText` is invalid.

Implements

[IDbCommand.Prepare](#)

See also

- ◆ [“ULCommand class” on page 57](#)

- ◆ “ULCommand members” on page 58
- ◆ “Connection property” on page 65
- ◆ “Transaction property” on page 68
- ◆ “CommandText property” on page 63

ULCommandBuilder class

Automatically generates single-table commands used to reconcile changes made to a System.Data.DataSet with the associated database.

Syntax

Visual Basic

NotInheritable Public Class **ULCommandBuilder**
Inherits Component

C#

```
public sealed class ULCommandBuilder :  
    Component
```

Remarks

The ULDataAdapter does not automatically generate the SQL statements required to reconcile changes made to a System.Data.DataSet with the associated data source. However, you can create a ULCommandBuilder object to automatically generate SQL statements for single-table updates if you set the SelectCommand property of the ULDataAdapter. Then, any additional SQL statements that you do not set are generated by the ULCommandBuilder.

Inherits: System.ComponentModel.Component

Implements: System.IDisposable

See also

- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [DataSet](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [System.ComponentModel.Component](#)
- ◆ [IDisposable](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULConnection class” on page 90](#)

ULCommandBuilder members

Public constructors

Member name	Description
ULCommandBuilder constructor	Initializes a ULCommandBuilder object.
ULCommandBuilder constructor	Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

Public properties

Member name	Description
DataAdapter property	Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.
QuotePrefix property	Gets or sets the starting character or characters to use when specifying UltraLite database objects, such as tables or columns, whose names contain characters such as spaces or reserved tokens.
QuoteSuffix property	Gets or sets the ending character or characters to use when specifying UltraLite database objects, such as tables or columns, whose names contain characters such as spaces or reserved tokens.

Public methods

Member name	Description
GetDeleteCommand method	Gets the automatically generated ULCommand object required to perform deletions on the database.
GetInsertCommand method	Gets the automatically generated ULCommand object required to perform insertions on the database.
GetUpdateCommand method	Gets the automatically generated ULCommand object required to perform updates on the database.
RefreshSchema method	Clears the commands associated with this ULCommandBuilder.

ULCommandBuilder constructor

Initializes a ULCommandBuilder object.

Syntax**Visual Basic**

Overloads Public Sub **New()**

C#

```
public ULCommandBuilder();
```

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“ULCommandBuilder constructor” on page 84](#)

ULCommandBuilder constructor

Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal adapter As ULDataAdapter _  
)
```

C#

```
public ULCommandBuilder(  
    ULDataAdapter adapter  
);
```

Parameters

- ♦ **adapter** A ULDataAdapter object.

See also

- ♦ [“ULCommandBuilder class” on page 82](#)
- ♦ [“ULCommandBuilder members” on page 82](#)
- ♦ [“ULDataAdapter class” on page 153](#)

DataAdapter property

Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.

Syntax

Visual Basic

```
Public Property DataAdapter As ULDataAdapter
```

C#

```
public ULDataAdapter DataAdapter {get;set;}
```

Property value

A ULDataAdapter object.

See also

- ♦ [“ULCommandBuilder class” on page 82](#)
- ♦ [“ULCommandBuilder members” on page 82](#)
- ♦ [“ULDataAdapter class” on page 153](#)

QuotePrefix property

Gets or sets the starting character or characters to use when specifying UltraLite database objects, such as tables or columns, whose names contain characters such as spaces or reserved tokens.

Syntax

Visual Basic

Public Property **QuotePrefix** As String

C#

public string **QuotePrefix** {get;set;}

Property value

The starting character or characters to use. The default is an empty string.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“QuoteSuffix property” on page 85](#)

QuoteSuffix property

Gets or sets the ending character or characters to use when specifying UltraLite database objects, such as tables or columns, whose names contain characters such as spaces or reserved tokens.

Syntax

Visual Basic

Public Property **QuoteSuffix** As String

C#

public string **QuoteSuffix** {get;set;}

Property value

The ending character or characters to use. The default is an empty string.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“QuotePrefix property” on page 85](#)

GetDeleteCommand method

Gets the automatically generated ULCommand object required to perform deletions on the database.

Syntax

Visual Basic

Public Function **GetDeleteCommand()** As ULCommand

C#

public ULCommand **GetDeleteCommand();**

Return value

The automatically generated ULCommand object required to perform deletions.

Remarks

After the SQL statement is first generated, the application must explicitly call the RefreshSchema if it changes the ULDataAdapter.SelectCommand in any way. Otherwise, the GetDeleteCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the ULDataAdapter.Update or the GetDeleteCommand method.

Exceptions

- ◆ [InvalidOperationException](#) - The ULCommandBuilder.DataAdapter has not been initialized.

The DataAdapter.SelectCommand property has not been initialized.

The DataAdapter.SelectCommand.Connection property has not been initialized.

Dynamic SQL generation is not supported against multiple base tables.

Dynamic SQL generation is not supported against a SelectCommand that contains duplicate columns.

Dynamic SQL generation for the DeleteCommand is not supported against a SelectCommand that does not return any key column information.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“RefreshSchema method” on page 88](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“Update method” on page 172](#)
- ◆ [“DataAdapter property” on page 84](#)

GetInsertCommand method

Gets the automatically generated ULCommand object required to perform insertions on the database.

Syntax**Visual Basic**

Public Function **GetInsertCommand()** As ULCommand

C#

public ULCommand **GetInsertCommand();**

Return value

The automatically generated ULCommand object required to perform insertions.

Remarks

After the SQL statement is first generated, the application must explicitly call the RefreshSchema if it changes the ULDataAdapter.SelectCommand in any way. Otherwise, the GetInsertCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the ULDataAdapter.Update or the GetInsertCommand method.

Exceptions

- ◆ [InvalidOperationException](#) - The ULCommandBuilder.DataAdapter has not been initialized.

The DataAdapter.SelectCommand property has not been initialized.

The DataAdapter.SelectCommand.Connection property has not been initialized.

Dynamic SQL generation for the InsertCommand is not supported against a SelectCommand that does not return any modifiable columns.

Dynamic SQL generation is not supported against multiple base tables.

Dynamic SQL generation is not supported against a SelectCommand that contains duplicate columns.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“RefreshSchema method” on page 88](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“DataAdapter property” on page 84](#)

GetUpdateCommand method

Gets the automatically generated ULCommand object required to perform updates on the database.

Syntax

Visual Basic

Public Function **GetUpdateCommand()** As ULCommand

C#

public ULCommand **GetUpdateCommand();**

Return value

The automatically generated ULCommand object required to perform updates.

Remarks

After the SQL statement is first generated, the application must explicitly call the RefreshSchema if it changes the ULDataAdapter.SelectCommand in any way. Otherwise, the GetUpdateCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the ULDataAdapter.Update or the GetUpdateCommand method.

Exceptions

- ◆ [InvalidOperationException](#) - The ULCommandBuilder.DataAdapter has not been initialized.

The DataAdapter.SelectCommand property has not been initialized.

The DataAdapter.SelectCommand.Connection property has not been initialized.

Dynamic SQL generation for the UpdateCommand is not supported against a SelectCommand that does not return any modifiable columns.

Dynamic SQL generation is not supported against multiple base tables.

Dynamic SQL generation is not supported against a SelectCommand that contains duplicate columns.

Dynamic SQL generation for the UpdateCommand is not supported against a SelectCommand that does not return any key column information.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“RefreshSchema method” on page 88](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“Update method” on page 172](#)
- ◆ [“DataAdapter property” on page 84](#)

RefreshSchema method

Clears the commands associated with this ULCommandBuilder.

Syntax**Visual Basic**

```
Public Sub RefreshSchema()
```

C#

```
public void RefreshSchema();
```

Remarks

After the SQL statement is first generated, the application must explicitly call RefreshSchema if it changes the ULDataAdapter.SelectCommand statement in any way. Otherwise, GetInsertCommand, GetDeleteCommand, and GetUpdateCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either ULDataAdapter.Update, GetInsertCommand, GetDeleteCommand, or GetUpdateCommand.

See also

- ◆ [“ULCommandBuilder class” on page 82](#)
- ◆ [“ULCommandBuilder members” on page 82](#)
- ◆ [“RefreshSchema method” on page 88](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“GetInsertCommand method” on page 86](#)
- ◆ [“Update method” on page 172](#)
- ◆ [“GetInsertCommand method” on page 86](#)
- ◆ [“GetDeleteCommand method” on page 85](#)
- ◆ [“GetUpdateCommand method” on page 87](#)

ULConnection class

Represents a connection to an UltraLite.NET database.

Syntax

Visual Basic

NotInheritable Public Class **ULConnection**
Inherits Component

C#

public sealed class **ULConnection** :
Component

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set `ULDatabaseManager.RuntimeType` to the appropriate value before using any other UltraLite.NET API.

A connection to an existing database is opened using the `ULConnection.Open`.

You must open a connection before carrying out any other operation, and you must close the connection after you have finished all operations on the connection and before your application terminates. In addition, you must close all result sets and tables opened on a connection before closing the connection.

The schema of the database can be accessed using an open connection's `ULConnection.Schema`.

Implements: `System.Data.IDbConnection`, `System.IDisposable`

See also

- ◆ [“ULConnection members” on page 90](#)
- ◆ [“RuntimeType property” on page 179](#)
- ◆ [“Open method” on page 115](#)
- ◆ [“Schema property” on page 100](#)
- ◆ [IDbConnection](#)
- ◆ [IDisposable](#)

ULConnection members

Public static fields (shared)

Member name	Description
INVALID_DATABASE_ID field	UL Ext.: A database ID constant indicating that the <code>ULConnection.DatabaseID</code> has not been set.

Public static properties (shared)

Member name	Description
DatabaseManager property	UL Ext.: Provides access to the singleton ULDatabaseManager object.

Public constructors

Member name	Description
ULConnection constructor	Initializes a ULConnection object. The connection must be opened before you can perform any operations against the database.
ULConnection constructor	Initializes a ULConnection object with the specified connection string. The connection must be opened before you can perform any operations against the database.

Public properties

Member name	Description
ConnectionString property	Specifies the parameters to use for opening a connection to an UltraLite.NET database. The connection string can be supplied using a ULConnectionParms object.
ConnectionTimeout property	This feature is not supported by UltraLite.NET.
Database property	Returns the name of the database to which the connection opens.
DatabaseID property	UL Ext.: Specifies the Database ID value to be used for global autoincrement columns.
GlobalAutoIncrementUsage property	UL Ext.: Returns the percentage of available global autoincrement values that have been used.
LastIdentity property	UL Ext.: Returns the most recent identity value used.
Schema property	UL Ext.: Provides access to the schema of the current database associated with this connection.
State property	Returns the current state of the connection.
SyncParms property	UL Ext.: Specifies the synchronization settings for this connection.
SyncResult property	UL Ext.: Returns the results of the last synchronization for this connection.

Public methods

Member name	Description
BeginTransaction method	Returns a transaction object. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with <code>ULTransaction.Commit</code> or <code>ULTransaction.Rollback</code> .
BeginTransaction method	Returns a transaction object with the specified isolation level. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with <code>ULTransaction.Commit</code> or <code>ULTransaction.Rollback</code> .
ChangeDatabase method	Changes the current database for an open <code>ULConnection</code> .
ChangeEncryptionKey method	UL Ext.: Changes the database's encryption key to the specified new key.
Close method	Closes the database connection.
CountUploadRows method	UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.
CountUploadRows method	UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.
CreateCommand method	Creates and initializes a <code>ULCommand</code> object associated with this connection and its current transaction. You can use the properties of the <code>ULCommand</code> to control its behavior.
ExecuteTable method	UL Ext.: Retrieves in a <code>ULTable</code> a database table for direct manipulation. The table is opened (sorted) using the table's primary key.
ExecuteTable method	UL Ext.: Retrieves in a <code>ULTable</code> a database table for direct manipulation. The table is opened (sorted) using the specified index.
ExecuteTable method	UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The table is opened (sorted) using the specified index.
GetLastDownloadTime method	UL Ext.: Returns the time of the most recent download of the specified publication.
GetNewUUID method	UL Ext.: Generates a new UUID (<code>System.Guid</code>).
GrantConnectTo method	UL Ext.: Grants access to an UltraLite database for a user ID with a specified password.
Open method	Opens a connection to a database using the previously-specified connection string.
ResetLastDownloadTime method	UL Ext.: Resets the time of the most recent download.
RevokeConnectFrom method	UL Ext.: Revokes access to an UltraLite database from the specified user ID.

Member name	Description
RollbackPartialDownload method	UL Ext.: Rolls back outstanding changes to the database from a partial download.
StartSynchronizationDelete method	UL Ext.: Marks all subsequent deletes made by this connection for synchronization.
StopSynchronizationDelete method	UL Ext.: Prevents delete operations from being synchronized.
Synchronize method	UL Ext.: Synchronize the database using the current ULConnection.SyncParms.
Synchronize method	UL Ext.: Synchronize the database using the current ULConnection.SyncParms with progress events posted to the specified listener.

Public events

Member name	Description
InfoMessage event	Occurs when UltraLite.NET sends a warning or an informational message on this connection.
StateChange event	Occurs when this connection changes state.

ULConnection constructor

Initializes a ULConnection object. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

Overloads Public Sub **New()**

C#

```
public ULConnection();
```

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set ULDatabaseManager.RuntimeType to the appropriate value before using any other UltraLite.NET API.

The ULConnection object needs to have the ULConnection.ConnectionString set before it can be opened.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)

- ◆ [“Open method” on page 115](#)
- ◆ [“ConnectionString property” on page 95](#)

ULConnection constructor

Initializes a ULConnection object with the specified connection string. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal connectionString As String _  
)
```

C#

```
public ULConnection(  
    string connectionString  
);
```

Parameters

- ◆ **connectionString** An UltraLite.NET connection string. A connection string is a semicolon-separated list of keyword-value pairs.

For a list of parameters, see the [ConnectionString property](#).

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set ULDatabaseManager.RuntimeType to the appropriate value before using any other UltraLite.NET API.

The connection string can be supplied using a ULConnectionParms object.

Exceptions

- ◆ [ArgumentException](#) - The supplied connection string is invalid.

Example

The following code creates and opens a connection to the existing database \UltraLite\MyDatabase.udb on a Windows CE device.

```
' Visual Basic  
Dim openParms As ULConnectionParms = New ULConnectionParms  
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"  
Dim conn As ULConnection = _  
    New ULConnection( openParms.ToString() )  
conn.Open()  
  
// C#  
ULConnectionParms openParms = new ULConnectionParms();  
openParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";  
ULConnection conn = new ULConnection( openParms.ToString() );  
conn.Open();
```


See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“Open method” on page 115](#)
- ◆ [“ULConnection constructor” on page 93](#)
- ◆ [“RuntimeType property” on page 179](#)
- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“ConnectionString property” on page 95](#)

INVALID_DATABASE_ID field

UL Ext.: A database ID constant indicating that the ULConnection.DatabaseID has not been set.

Syntax**Visual Basic**

```
Public Shared INVALID_DATABASE_ID As Long
```

C#

```
public const long INVALID_DATABASE_ID;
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“DatabaseID property” on page 97](#)

ConnectionString property

Specifies the parameters to use for opening a connection to an UltraLite.NET database. The connection string can be supplied using a ULConnectionParms object.

Syntax**Visual Basic**

```
NotOverridable Public Property ConnectionString As String _  
    Implements IDbConnection.ConnectionString
```

C#

```
public string ConnectionString {get;set;}
```

Property value

The parameters used to open this connection in the form of a semicolon-separated list of keyword-value pairs. The default is an empty string (an invalid connection string).

Remarks

UL Ext.: The parameters used by UltraLite.NET are specific to UltraLite databases and therefore the connection string is not compatible with SQL Anywhere connection strings. For a list of parameters, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

Parameter values must not contain semi-colons (;), or begin with either a single quote (') or a double quote (") character. Leading and trailing spaces in values are ignored.

By default, connections are opened with UID=DBA and PWD=sql. To make the database more secure, change the user DBA's password or create new users (using `GrantConnectTo`) and remove the DBA user (using `RevokeConnectFrom`).

Exceptions

- ◆ [ArgumentException](#) - The supplied connection string is invalid.
- ◆ [InvalidOperationException](#) - The value cannot be set while the connection is open.

Implements

[IDbConnection.ConnectionString](#)

Example

The following code creates and opens a connection to the existing database \UltraLite\MyDatabase.udb on a Windows CE device.

```
' Visual Basic
Dim openParms As ULConnectionParms = New ULConnectionParms
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"
Dim conn As ULConnection = New ULConnection
conn.ConnectionString = openParms.ToString()
conn.Open()

// C#
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";
ULConnection conn = new ULConnection();
conn.ConnectionString = openParms.ToString();
conn.Open();
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“Open method” on page 115](#)
- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“GrantConnectTo method” on page 114](#)

ConnectionTimeout property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

NotOverridable Public Readonly Property **ConnectionTimeout** As Integer _
Implements IDbConnection.ConnectionTimeout

C#

public int **ConnectionTimeout** {get;}

Property value

The value is always zero.

Exceptions

- ◆ [ULException class](#) - Setting the value is not supported in UltraLite.NET.

Implements

[IDbConnection.ConnectionTimeout](#)

Database property

Returns the name of the database to which the connection opens.

Syntax

Visual Basic

NotOverridable Public Readonly Property **Database** As String _
Implements IDbConnection.Database

C#

public string **Database** {get;}

Property value

A string containing the name of the database.

Remarks

On Windows CE devices, ULConnection looks in the connection string in the following order: dbn, ce_file.

On desktop machines, ULConnection looks in the connection string in the following order: dbn, nt_file.

Implements

[IDbConnection.Database](#)

DatabaselD property

UL Ext.: Specifies the Database ID value to be used for global autoincrement columns.

Syntax

Visual Basic

Public Property **DatabaseID** As Long

C#

```
public long DatabaseID {get;set;}
```

Property value

The Database ID value of the current database.

Remarks

The database ID value must be in the range [0, System.UInt32.MaxValue]. A value of `ULConnection.INVALID_DATABASE_ID` is used to indicate that the database ID has not been set for the current database.

Exceptions

- ◆ [ULException class](#) - The specified new database ID is invalid.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“GetDatabaseProperty method” on page 188](#)
- ◆ [“SetDatabaseOption method” on page 193](#)
- ◆ [UInt32.MaxValue](#)
- ◆ [“INVALID_DATABASE_ID field” on page 95](#)

DatabaseManager property

UL Ext.: Provides access to the singleton `ULDatabaseManager` object.

Syntax

Visual Basic

Public Shared Readonly Property **DatabaseManager** As `ULDatabaseManager`

C#

```
public const ULDatabaseManager DatabaseManager {get;}
```

Property value

A reference to the singleton `ULDatabaseManager` object.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULDatabaseManager class” on page 178](#)

GlobalAutoIncrementUsage property

UL Ext.: Returns the percentage of available global autoincrement values that have been used.

Syntax

Visual Basic

Public Readonly Property **GlobalAutoIncrementUsage** As Short

C#

```
public short GlobalAutoIncrementUsage {get;}
```

Property value

The percentage of available global autoincrement values that have been used. It is an integer in the range [0-100], inclusive.

Remarks

If the percentage approaches 100, your application should set a new value for the global database ID using `ULConnection.DatabaseID`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“DatabaseID property” on page 97](#)

LastIdentity property

UL Ext.: Returns the most recent identity value used.

Syntax

Visual Basic

Public Readonly Property **LastIdentity** As UInt64

C#

```
public ulong LastIdentity {get;}
```

Property value

The most recently-used identity value as an unsigned long.

Remarks

The most recent identity value used. This property is equivalent to the SQL Anywhere statement:

```
SELECT @@identity
```

LastIdentity is particularly useful in the context of global autoincrement columns.

Since this property only allows you to determine the most recently assigned default value, you should retrieve this value soon after executing the insert statement to avoid spurious results.

Occasionally, a single insert statement may include more than one column of type global autoincrement. In this case, LastIdentity is one of the generated default values, but there is no reliable means to determine from which column the value is. For this reason, you should design your database and write your insert statements to avoid this situation.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Schema property

UL Ext.: Provides access to the schema of the current database associated with this connection.

Syntax

Visual Basic

Public Readonly Property **Schema** As ULDatabaseSchema

C#

```
public ULDatabaseSchema Schema {get;}
```

Property value

A reference to the ULDatabaseSchema object representing the schema of the database on which this connection opens.

Remarks

This property is only valid while its connection is open.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULDatabaseSchema class” on page 185](#)

State property

Returns the current state of the connection.

Syntax

Visual Basic

NotOverridable Public Readonly Property **State** As ConnectionState _
Implements IDbConnection.State

C#

```
public ConnectionState State {get;}
```

Property value

System.Data.ConnectionState.Open if the connection is open, System.Data.ConnectionState.Closed if the connection is closed.

Implements

[IDbConnection.State](#)

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“StateChange event” on page 120](#)
- ◆ [ConnectionState.Open](#)
- ◆ [ConnectionState.Closed](#)

SyncParms property

UL Ext.: Specifies the synchronization settings for this connection.

Syntax

Visual Basic

Public Readonly Property **SyncParms** As ULSyncParms

C#

```
public ULSyncParms SyncParms {get;}
```

Property value

A reference to the ULSyncParms object representing the parameters used for synchronization by this connection. Modifications to the parameters affect the next synchronization made over this connection.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“SyncResult property” on page 102](#)
- ◆ [“ULSyncParms class” on page 385](#)

SyncResult property

UL Ext.: Returns the results of the last synchronization for this connection.

Syntax

Visual Basic

Public Readonly Property **SyncResult** As ULSyncResult

C#

public ULSyncResult **SyncResult** {get;}

Property value

A reference to the ULSyncResult object representing the results of the last synchronization for this connection.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“SyncParms property” on page 101](#)
- ◆ [“SyncResult property” on page 102](#)

BeginTransaction method

Returns a transaction object. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with ULTransaction.Commit or ULTransaction.Rollback.

Syntax

Visual Basic

Overloads Public Function **BeginTransaction()** As ULTransaction

C#

public ULTransaction **BeginTransaction();**

Return value

A ULTransaction object representing the new transaction.

Remarks

To associate a command with a transaction object, use the ULCommand.Transaction property. The current transaction is automatically associated to commands created by ULConnection.CreateCommand.

By default, the connection does not use transactions and all commands are automatically committed as they are executed. Once the current transaction is committed or rolled back, the connection reverts to auto commit mode until the next call to BeginTransaction.

This is the strongly-typed version of `System.Data.IDbConnection.BeginTransaction()`.

Exceptions

- ◆ [ULException class](#) - The connection is closed.
- ◆ [InvalidOperationException](#) - ULConnection does not support parallel transactions.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“BeginTransaction method” on page 103](#)
- ◆ [“Commit method” on page 458](#)
- ◆ [“Rollback method” on page 459](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [IDbConnection.BeginTransaction](#)

BeginTransaction method

Returns a transaction object with the specified isolation level. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with `ULTransaction.Commit` or `ULTransaction.Rollback`.

Syntax

Visual Basic

```
Overloads Public Function BeginTransaction( _  
    ByVal isolationLevel As IsolationLevel _  
) As ULTransaction
```

C#

```
public ULTransaction BeginTransaction(  
    IsolationLevel isolationLevel  
);
```

Parameters

- ◆ **isolationLevel** The required isolation level for the transaction. UltraLite.NET only supports `System.Data.IsolationLevel.ReadUncommitted`.

Return value

A `ULTransaction` object representing the new transaction.

Remarks

To associate a command with a transaction object, use the `ULCommand.Transaction` property. The current transaction is automatically associated to commands created by `ULConnection.CreateCommand`.

By default, the connection does not use transactions and all commands are automatically committed as they are executed. Once the current transaction is committed or rolled back, the connection reverts to auto commit mode until the next call to `BeginTransaction`.

This is the strongly-typed version of `System.Data.IDbConnection.BeginTransaction` (`System.Data.IsolationLevel`).

Exceptions

- ◆ [ULException class](#) - The connection is closed or an unsupported isolation level was specified.
- ◆ [InvalidOperationException](#) - `ULConnection` does not support parallel transactions.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“BeginTransaction method” on page 102](#)
- ◆ [“Commit method” on page 458](#)
- ◆ [“Rollback method” on page 459](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [“CreateCommand method” on page 108](#)
- ◆ [IDbConnection.BeginTransaction](#)
- ◆ [IsolationLevel.ReadUncommitted](#)

ChangeDatabase method

Changes the current database for an open `ULConnection`.

Syntax

Visual Basic

```
NotOverridable Public Sub ChangeDatabase( _  
    ByVal connectionString As String _  
) _  
    Implements IDbConnection.ChangeDatabase
```

C#

```
public void ChangeDatabase(  
    string connectionString  
);
```

Parameters

- ◆ **connectionString** A complete connection string to open the connection to a new database.

Remarks

The connection to the current database is closed even if there are parameter errors.

UL Ext.: *connectionString* is a full connection string, not a dbn or dbf.

Implements

[IDbConnection.ChangeDatabase](#)

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ConnectionString property” on page 95](#)

ChangeEncryptionKey method

UL Ext.: Changes the database's encryption key to the specified new key.

Syntax

Visual Basic

```
Public Sub ChangeEncryptionKey( _  
    ByVal newKey As String _  
)
```

C#

```
public void ChangeEncryptionKey(  
    string newKey  
);
```

Parameters

- ◆ **newKey** The new encryption key for the database.

Remarks

If the encryption key is lost, it is not possible to open the database.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“EncryptionKey property” on page 130](#)

Close method

Closes the database connection.

Syntax

Visual Basic

NotOverridable Public Sub **Close()** _
Implements IDbConnection.Close

C#

public void **Close()**;

Remarks

The Close method rolls back any pending transactions and then closes the connection. An application can call Close multiple times.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDbConnection.Close](#)

CountUploadRows method

UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.

Syntax

Visual Basic

Overloads Public Function **CountUploadRows**(_
ByVal *mask* As Integer, _
ByVal *threshold* As UInt32 _
) As UInt32

C#

public uint **CountUploadRows**(
int *mask*,
uint *threshold*
);

Parameters

- ◆ **mask** The set of publications to check for rows. For more information, see the [ULPublicationSchema class](#).
- ◆ **threshold** The maximum number of rows to count, limiting the amount of time taken by CountUploadRows. A value of 0 corresponds to the maximum limit. A value of 1 determines if any rows need to be synchronized.

Return value

The number of rows that need to be uploaded from the specified publication(s).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“CountUploadRows method” on page 107](#)

CountUploadRows method

UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.

Syntax**Visual Basic**

```
Overloads Public Function CountUploadRows( _  
    ByVal mask As Integer, _  
    ByVal threshold As Long _  
) As Long
```

C#

```
public long CountUploadRows(  
    int mask,  
    long threshold  
);
```

Parameters

- ◆ **mask** The set of publications to check for rows. For more information, see the [ULPublicationSchema class](#).
- ◆ **threshold** The maximum number of rows to count, limiting the amount of time taken by CountUploadRows. A value of 0 corresponds to the maximum limit. A value of 1 determines if any rows need to be synchronized. The threshold value must be in the range [0,0xffffffff].

Return value

The number of rows that need to be uploaded from the specified publication(s).

Remarks

This method is provided for languages that do not support the System.UInt32 type natively. Use the other form of this method if your application supports it.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“CountUploadRows method” on page 106](#)

CreateCommand method

Creates and initializes a ULCommand object associated with this connection and its current transaction. You can use the properties of the ULCommand to control its behavior.

Syntax

Visual Basic

```
Public Function CreateCommand() As ULCommand
```

C#

```
public ULCommand CreateCommand();
```

Return value

A new ULCommand object.

Remarks

You must set the ULCommand.CommandText before the command can be executed.

This is the strongly-typed version of System.Data.IDbConnection.CreateCommand.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [IDbConnection.CreateCommand](#)

ExecuteTable method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The table is opened (sorted) using the table's primary key.

Syntax

Visual Basic

```
Overloads Public Function ExecuteTable( _  
    ByVal tableName As String _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    string tableName  
);
```

Parameters

- ◆ **tableName** The name of the table to open.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the ULCommand.ExecuteTable() method that does not require a ULCommand instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces iAnywhere.UltraLite.Connection.GetTable() and iAnywhere.UltraLite.Table.Open()).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The *tableName* is invalid.

Example

The following code opens the table named MyTable using the table's primary key. It assumes an open ULConnection instance called conn.

```
' Visual Basic
Dim t As ULTable = conn.ExecuteTable("MyTable")
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
' cmd.CommandText = "MyTable"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable = cmd.ExecuteTable()
' cmd.Dispose()

// C#
ULTable t = conn.ExecuteTable("MyTable");
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable();
// }
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ExecuteTable method” on page 109](#)
- ◆ [“ExecuteTable method” on page 111](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULCommand class” on page 57](#)

ExecuteTable method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The table is opened (sorted) using the specified index.

Syntax

Visual Basic

Overloads Public Function **ExecuteTable**(_
 ByVal *tableName* As String, _
 ByVal *indexName* As String _
) As ULTable

C#

```
public ULTable ExecuteTable(  
    string tableName,  
    string indexName  
);
```

Parameters

- ◆ **tableName** The name of the table to open.
- ◆ **indexName** The name of the index with which to open (sort) the table.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the ULCommand.ExecuteTable() method that does not require a ULCommand instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces iAnywhere.UltraLite.Connection.GetTable() and iAnywhere.UltraLite.Table.Open()).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The *tableName* is invalid.

Example

The following code opens the table named MyTable using the index named MyIndex. It assumes an open ULConnection instance called conn.

```
' Visual Basic  
Dim t As ULTable = conn.ExecuteTable("MyTable", "MyIndex")  
' The line above is equivalent to  
' Dim cmd As ULCommand = conn.CreateCommand()  
' cmd.CommandText = "MyTable"  
' cmd.IndexName = "MyIndex"  
' cmd.CommandType = CommandType.TableDirect  
' Dim t As ULTable = cmd.ExecuteTable()  
' cmd.Dispose()  
  
// C#  
ULTable t = conn.ExecuteTable("MyTable", "MyIndex");  
// The line above is equivalent to  
// ULTable t;  
// using(ULCommand cmd = conn.CreateCommand())  
// {  
//     cmd.CommandText = "MyTable";  
//     cmd.IndexName = "MyIndex";
```



```
//      cmd.CommandType = CommandType.TableDirect;  
//      t = cmd.ExecuteTable();  
// }
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ExecuteTable method” on page 108](#)
- ◆ [“ExecuteTable method” on page 111](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULCommand class” on page 57](#)

ExecuteTable method

UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The table is opened (sorted) using the specified index.

Syntax

Visual Basic

Overloads Public Function **ExecuteTable**(_
 ByVal *tableName* As String, _
 ByVal *indexName* As String, _
 ByVal *cmdBehavior* As CommandBehavior _
) As ULTable

C#

```
public ULTable ExecuteTable(  
    string tableName,  
    string indexName,  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **tableName** The name of the table to open.
- ◆ **indexName** The name of the index with which to open (sort) the table.
- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the `ULCommand.ExecuteTable(System.Data.CommandBehavior)` that does not require a `ULCommand` instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces `iAnywhere.UltraLite.Connection.GetTable()` and `iAnywhere.UltraLite.Table.Open()`).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [InvalidOperationException](#) - The *tableName* is invalid.

Example

The following code opens the table named `MyTable` using the index named `MyIndex`. It assumes an open `ULConnection` instance called `conn`.

```
' Visual Basic
Dim t As ULTable = conn.ExecuteTable( _
    "MyTable", "MyIndex", CommandBehavior.Default _
)
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
' cmd.CommandText = "MyTable"
' cmd.IndexName = "MyIndex"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable = cmd.ExecuteTable(CommandBehavior.Default)
' cmd.Dispose()

// C#
ULTable t = conn.ExecuteTable(
    "MyTable", "MyIndex", CommandBehavior.Default
);
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.IndexName = "MyIndex";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable(CommandBehavior.Default);
// }
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ExecuteTable method” on page 108](#)
- ◆ [“ExecuteTable method” on page 109](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“ExecuteTable method” on page 79](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)

GetLastDownloadTime method

UL Ext.: Returns the time of the most recent download of the specified publication.

Syntax

Visual Basic

```
Public Function GetLastDownloadTime( _  
    ByVal mask As Integer _  
) As Date
```

C#

```
public DateTime GetLastDownloadTime(  
    int mask  
);
```

Parameters

- ◆ **mask** The mask of the publication to check. For more information, see the [ULPublicationSchema class](#).

Return value

The timestamp of the last download.

Remarks

The parameter *mask* must reference a single publication or be the special constant `ULPublicationSchema.SYNC_ALL_DB` for the time of the last download of the full database.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ResetLastDownloadTime method” on page 115](#)
- ◆ [“SYNC_ALL_DB field” on page 313](#)

GetNewUUID method

UL Ext.: Generates a new UUID (System.Guid).

Syntax

Visual Basic

```
Public Function GetNewUUID() As Guid
```

C#

```
public Guid GetNewUUID();
```

Return value

A new UUID as a System.Guid.

Remarks

This method is provided here because it is not included in the .NET Compact Framework.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [Guid](#)

GrantConnectTo method

UL Ext.: Grants access to an UltraLite database for a user ID with a specified password.

Syntax**Visual Basic**

```
Public Sub GrantConnectTo( _  
    ByVal uid As String, _  
    ByVal pwd As String _  
)
```

C#

```
public void GrantConnectTo(  
    string uid,  
    string pwd  
);
```

Parameters

- ◆ **uid** The user ID to receive access to the database. The maximum length of the user ID is 16 characters.
- ◆ **pwd** The password to be associated with the user ID. The maximum length is 16 characters.

Remarks

If an existing user ID is specified, this function updates the password for the user. UltraLite supports a maximum of 4 users. This method is enabled only if user authentication was enabled when the connection was opened.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“UserID property” on page 131](#)
- ◆ [“Password property” on page 130](#)

- ◆ [“ConnectionString property” on page 95](#)

Open method

Opens a connection to a database using the previously-specified connection string.

Syntax

Visual Basic

```
NotOverridable Public Sub Open() _  
    Implements IDbConnection.Open
```

C#

```
public void Open();
```

Remarks

You should explicitly close or dispose of the connection when you are done with it.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred while attempting to open the database.
- ◆ [InvalidOperationException](#) - The connection is already open or the connection string is not specified in `ULConnection.ConnectionString`.

Implements

[IDbConnection.Open](#)

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ConnectionString property” on page 95](#)
- ◆ [“State property” on page 100](#)

ResetLastDownloadTime method

UL Ext.: Resets the time of the most recent download.

Syntax

Visual Basic

```
Public Sub ResetLastDownloadTime( _  
    ByVal mask As Integer _  
)
```

C#

```
public void ResetLastDownloadTime(
```

```
int mask  
);
```

Parameters

- ◆ **mask** The set of publications to reset. For more information, see the [ULPublicationSchema class](#).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“GetLastDownloadTime method” on page 113](#)

RevokeConnectFrom method

UL Ext.: Revokes access to an UltraLite database from the specified user ID.

Syntax**Visual Basic**

```
Public Sub RevokeConnectFrom( _  
    ByVal uid As String _  
)
```

C#

```
public void RevokeConnectFrom(  
    string uid  
);
```

Parameters

- ◆ **uid** The user ID whose access to the database is being revoked.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“GrantConnectTo method” on page 114](#)

RollbackPartialDownload method

UL Ext.: Rolls back outstanding changes to the database from a partial download.

Syntax

Visual Basic

```
Public Sub RollbackPartialDownload()
```

C#

```
public void RollbackPartialDownload();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“KeepPartialDownload property” on page 389](#)
- ◆ [“ResumePartialDownload property” on page 392](#)

StartSynchronizationDelete method

UL Ext.: Marks all subsequent deletes made by this connection for synchronization.

Syntax

Visual Basic

```
Public Sub StartSynchronizationDelete()
```

C#

```
public void StartSynchronizationDelete();
```

Remarks

When this function is called, all delete operations are again synchronized, causing the rows deleted from the UltraLite database to be removed from the consolidated database as well.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“StopSynchronizationDelete method” on page 117](#)
- ◆ [“Truncate method” on page 439](#)

StopSynchronizationDelete method

UL Ext.: Prevents delete operations from being synchronized.

Syntax**Visual Basic**

Public Sub **StopSynchronizationDelete()**

C#

public void **StopSynchronizationDelete();**

Remarks

This method is useful for deleting old information on an UltraLite database to save space, while not deleting this information on the consolidated database.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“StartSynchronizationDelete method” on page 117](#)

Synchronize method

UL Ext.: Synchronize the database using the current ULConnection.SyncParms.

Syntax**Visual Basic**

Overloads Public Sub **Synchronize()**

C#

public void **Synchronize();**

Remarks

A detailed result status is reported in this connection's ULConnection.SyncResult.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“Synchronize method” on page 119](#)
- ◆ [“SyncParms property” on page 101](#)
- ◆ [“SyncResult property” on page 102](#)

Synchronize method

UL Ext.: Synchronize the database using the current `ULConnection.SyncParms` with progress events posted to the specified listener.

Syntax

Visual Basic

```
Overloads Public Sub Synchronize( _  
    ByVal listener As ULSyncProgressListener _  
)
```

C#

```
public void Synchronize(  
    ULSyncProgressListener listener  
);
```

Parameters

- ◆ **listener** The object that receives synchronization progress events.

Remarks

The last event posted to the listener will have a state of `ULSyncProgressState.STATE_LAST`.

Errors during synchronization will be posted as `ULSyncProgressState.STATE_ERROR` events and then thrown as `ULExceptions`.

A detailed result status will be reported in this connection's `ULConnection.SyncResult`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULSyncProgressListener interface” on page 409](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“SyncParms property” on page 101](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“SyncResult property” on page 102](#)

InfoMessage event

Occurs when UltraLite.NET sends a warning or an informational message on this connection.

Syntax

Visual Basic

Public Event **InfoMessage** As ULInfoMessageEventHandler

C#

public event ULInfoMessageEventHandler **InfoMessage**;

Remarks

To process UltraLite.NET warnings or informational messages, you must create a ULInfoMessageEventHandler delegate and attach it to this event.

Example

The following code defines an informational message event handler.

```
' Visual Basic
Private Sub MyInfoMessageHandler( _
    obj As Object, args As ULInfoMessageEventArgs _
)
    System.Console.WriteLine( _
        "InfoMessageHandler: " + args.NativeError + ", " _
        + args.Message _
    )
End Sub

// C#
private void MyInfoMessageHandler(
    object obj, ULInfoMessageEventArgs args
)
{
    System.Console.WriteLine(
        "InfoMessageHandler: " + args.NativeError + ", "
        + args.Message
    );
}
```

The following code adds the MyInfoMessageHandler to the connection named conn.

```
' Visual Basic
AddHandler conn.InfoMessage, AddressOf MyInfoMessageHandler

// C#
conn.InfoMessage +=
    new ULInfoMessageEventHandler(MyInfoMessageHandler);
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [“ULInfoMessageEventHandler delegate” on page 276](#)

StateChange event

Occurs when this connection changes state.

Syntax

Visual Basic

Public Event **StateChange** As StateChangeEventHandler

C#

public event StateChangeEventHandler **StateChange**;

Remarks

To process state change messages, you must create a System.Data.StateChangeEventHandler delegate and attach it to this event.

Example

The following code defines a state change event handler.

```
' Visual Basic
Private Sub MyStateHandler( _
    obj As Object, args As StateChangeEventArgs _
)
    System.Console.WriteLine( _
        "StateHandler: " + args.OriginalState + " to " _
        + args.CurrentState _
    )
End Sub

// C#
private void MyStateHandler(
    object obj, StateChangeEventArgs args
)
{
    System.Console.WriteLine(
        "StateHandler: " + args.OriginalState + " to "
        + args.CurrentState
    );
}
```

The following code adds the MyStateHandler to the connection named conn.

```
' Visual Basic
AddHandler conn.StateChange, AddressOf MyStateHandler

// C#
conn.StateChange += new StateChangeEventHandler(MyStateHandler);
```

See also

- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection members” on page 90](#)
- ◆ [StateChangeEventHandler](#)

ULConnectionParms class

UL Ext.: Builds a connection string for opening a connection to an UltraLite database. The frequently-used connection parameters are individual properties on the ULConnectionParms object.

Syntax

Visual Basic

Public Class **ULConnectionParms**
Inherits Component

C#

```
public class ULConnectionParms :  
    Component
```

Remarks

A ULConnectionParms object is used to specify the parameters for opening a connection (ULConnection.Open) or dropping a database (ULDatabaseManager.DropDatabase).

Leading and trailing spaces are ignored in all values. Values must not contain leading or trailing spaces, or a semicolon (;), or begin with either a single quote (') or a double quote (").

When building a connection string, you need to identify the database and specify any optional connection settings. Once you have supplied all the connection parameters by setting the appropriate properties on a ULConnectionParms object, you create a connection string using the ULConnectionParms.ToString. The resulting string is used to create a new ULConnection with the ULConnection(String) constructor or set the ULConnection.ConnectionString of an existing ULConnection object.

Identifying the database

Each instance contains platform-specific paths to the database. Only the value corresponding to the executing platform is used. For example, in the code below the path \UltraLite\mydb1.udb would be used on Windows CE, while mydb2.db would be used on other platforms.

```
' Visual Basic  
Dim dbName As ULConnectionParms = new ULConnectionParms  
dbName.DatabaseOnCE = "\UltraLite\mydb1.udb"  
dbName.DatabaseOnDesktop = "somedir\mydb2.udb"  
  
// C#  
ULConnectionParms dbName = new ULConnectionParms();  
dbName.DatabaseOnCE = "\\UltraLite\\mydb1.udb";  
dbName.DatabaseOnDesktop = @"somedir\mydb2.udb";
```

The recommended extension for UltraLite database files is .udb. On Windows CE devices, the default database is \UltraLiteDB\ulstore.udb. On other Windows platforms, the default database is ulstore.udb. In C#, you must escape any backslash characters in paths or use @-quoted string literals.

If you are using multiple databases, you must specify a database name for each database. For more informaton, see the [AdditionalParms property](#).

Optional connection settings

Depending on your application's needs and how the database was created, you might need to supply a non-default `ULConnectionParms.UserID` and `ULConnectionParms.Password`, a database `ULConnectionParms.EncryptionKey`, and the connection `ULConnectionParms.CacheSize`. If your application is using multiple connections, you should provide a unique `ULConnectionParms.ConnectionName` for each connection.

Databases are created with a single authenticated user, DBA, whose initial password is sql. By default, connections are opened using the user ID DBA and password sql. To disable the default user, use the `ULConnection.RevokeConnectFrom`. To add a user or change a user's password, use the `ULConnection.GrantConnectTo`.

If an encryption key was supplied when the database was created, all subsequent connections to the database must use the same encryption key. To change a database's encryption key, use the `ULConnection.ChangeEncryptionKey`.

For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

See also

- ◆ [“ULConnectionParms members” on page 123](#)
- ◆ [“Open method” on page 115](#)
- ◆ [“DropDatabase method” on page 181](#)
- ◆ [“ToString method” on page 132](#)
- ◆ [“ULConnection class” on page 90](#)
- ◆ [“ULConnection constructor” on page 94](#)
- ◆ [“ConnectionString property” on page 95](#)
- ◆ [“UserID property” on page 131](#)
- ◆ [“Password property” on page 130](#)
- ◆ [“EncryptionKey property” on page 130](#)
- ◆ [“CacheSize property” on page 127](#)
- ◆ [“ConnectionName property” on page 128](#)
- ◆ [“RevokeConnectFrom method” on page 116](#)
- ◆ [“GrantConnectTo method” on page 114](#)
- ◆ [“ChangeEncryptionKey method” on page 105](#)

ULConnectionParms members

Public constructors

Member name	Description
ULConnectionParms constructor	Initializes a <code>ULConnectionParms</code> instance with its default values.

Public properties

Member name	Description
AdditionalParms property	Specifies additional parameters as a semicolon-separated list of name=value pairs. These are less commonly used parameters.

Member name	Description
CacheSize property	Specifies the size of the cache.
ConnectionString property	Specifies a name for the connection. This is only needed if you create more than one connection to the database.
DatabaseOnCE property	Specifies the path and file name of the UltraLite database on Windows CE.
DatabaseOnDesktop property	Specifies the path and file name of the UltraLite database on Windows desktop platforms.
EncryptionKey property	Specifies a key for encrypting the database.
Password property	Specifies the password for the authenticated user.
UserID property	Specifies an authenticated user for the database.

Public methods

Member name	Description
ToString method	Returns the string representation of this instance.

Public events

Member name	Description
UnusedEvent event	Unused.

ULConnectionParms constructor

Initializes a ULConnectionParms instance with its default values.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULConnectionParms();
```

AdditionalParms property

Specifies additional parameters as a semicolon-separated list of name=value pairs. These are less commonly used parameters.

Syntax

Visual Basic

Public Property **AdditionalParms** As String

C#

public string **AdditionalParms** {get;set;}

Property value

A semicolon-separated list of keyword=value additional parameters. Values of the keyword=value list must conform to the rules for ULConnection.ConnectionString. The default is a null reference (Nothing in Visual Basic).

Remarks

The values for the page size and reserve size parameters are specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix m or M to indicate megabytes.

Additional parameters are:

Keyword	Description
dbn	Identifies a loaded database to which a connection needs to be made. When a database is started, it is assigned a database name, either explicitly with the dbn parameter, or by UltraLite using the base of the file name with the extension and path removed. When opening connections, UltraLite first searches for a running database with a matching dbn. If one is not found, UltraLite starts a new database using the appropriate database file name parameter (DatabaseOnCE or DatabaseOnDesktop). This parameter is required if the application (or UltraLite engine) needs to access two different databases that have the same base file name. This parameter is only used when opening a connection with ULConnection.Open.

Keyword	Description
ordered_table_scans	Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default. As of release 10.0.1, when using dynamic SQL in UltraLite, if order is not important for executing a query, UltraLite will access the rows directly from the database pages rather than using the primary key index. This improves performance of fetching rows. To use this optimization, the query must be read only and must scan all the rows. When rows are expected in a specific order, an ORDER BY statement should be included as part of the SQL query. However, it's possible that some applications have come to rely on the behavior that defaults to returning rows in the primary key order. In this case, users should set the ordered_table_scans parameter to 1 (true, yes, on) to revert to the old behavior when iterating over a table. When ordered_table_scans is set to 1 (true, yes, on) and the user does not specify an ORDER BY clause or if a query would not benefit from an index, UltraLite will default to using the primary key. The following parameter string ensures that UltraLite behaves as in previous releases. <pre>createParms . AdditionalParms = "ordered_table_scans=yes"</pre> The default is 0 (false, off, no). This parameter is only used when opening a connection with <code>ULConnection.Open</code>. For more information, see “UltraLite Connection String Parameters Reference” [<i>UltraLite - Database Management and Reference</i>].

Keyword	Description
reserve_size	Reserves file system space for storage of UltraLite persistent data. The reserve_size parameter allows you to pre-allocate the file system space required for your UltraLite database without inserting any data. Reserving file system space can improve performance slightly and also prevent out of memory failures. By default, the persistent storage file only grows when required as the application updates the database. Note that reserve_size reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead, as well as data compression, must be considered when deriving the required file system space from the amount of database data. Running the database with test data and observing the persistent store file size is recommended. The reserve_size parameter reserves space by growing the persistent store file to the given reserve size on startup, regardless of whether the file previously existed. The file is never truncated. The following parameter string ensures that the persistent store file is at least 2 MB upon startup. createParms.AdditionalParms = "reserve_size=2m" This parameter is only used when opening a connection with ULConnection.Open.
start	Specifies the location and then starts the UltraLite engine. Only supply a StartLine (START) connection parameter if you are connecting to an engine that is not currently running. The location is only required when the UltraLite engine is not in the system path. For more information on using the UltraLite engine with UltraLite.NET, see the RuntimeType property.

Exceptions

- ◆ [ArgumentException](#) - The value contained an invalid connection string.

See also

- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“ULConnectionParms members” on page 123](#)
- ◆ [“ConnectionString property” on page 95](#)
- ◆ [“DatabaseOnCE property” on page 129](#)
- ◆ [“DatabaseOnDesktop property” on page 129](#)
- ◆ [“Open method” on page 115](#)

CacheSize property

Specifies the size of the cache.

Syntax

Visual Basic

Public Property **CacheSize** As String

C#

public string **CacheSize** {get;set;}

Property value

A string specifying the cache size. The default is a null reference (Nothing in Visual Basic) meaning the default of 16 pages is used.

Remarks

The values for the cache size are specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix of m or M to indicate megabytes.

For example, the following sets the cache size to 128 KB.

```
connParms.CacheSize = "128k"
```

The default cache size is 16 pages. Using the default page size of 4 KB, the default cache size is therefore 64 KB. The minimum cache size is platform dependent.

The default cache size is conservative. If your testing shows the need for better performance, you should increase the cache size.

Increasing the cache size beyond the size of the database itself provides no performance improvement and large cache sizes might interfere with the number of other applications you can use.

If the cache size is unspecified or improperly specified, the default size is used.

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

ConnectionString property

Specifies a name for the connection. This is only needed if you create more than one connection to the database.

Syntax

Visual Basic

Public Property **ConnectionString** As String

C#

public string **ConnectionString** {get;set;}

Property value

A string specifying the name of the connection. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

DatabaseOnCE property

Specifies the path and file name of the UltraLite database on Windows CE.

Syntax**Visual Basic**

Public Property **DatabaseOnCE** As String

C#

public string **DatabaseOnCE** {get;set;}

Property value

A string specifying the full path to the database. If the value is a null reference (Nothing in Visual Basic), the database \UltraLiteDB\ulstore.udb is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

DatabaseOnDesktop property

Specifies the path and file name of the UltraLite database on Windows desktop platforms.

Syntax**Visual Basic**

Public Property **DatabaseOnDesktop** As String

C#

public string **DatabaseOnDesktop** {get;set;}

Property value

A string specifying the absolute or relative path to the database. If the value is a null reference (Nothing in Visual Basic), the database ulstore.udb is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

EncryptionKey property

Specifies a key for encrypting the database.

Syntax

Visual Basic

Public Property **EncryptionKey** As String

C#

public string **EncryptionKey** {get;set;}

Property value

A string specifying the encryption key. The default is a null reference (Nothing in Visual Basic) meaning no encryption.

Remarks

All connections must use the same key as was specified when the database was created. Lost or forgotten keys result in completely inaccessible databases.

As with all passwords, it is best to choose a key value that cannot be easily guessed. The key can be of arbitrary length, but generally the longer the key, the better, because a shorter key is easier to guess than a longer one. Using a combination of numbers, letters, and special characters decreases the chances of someone guessing the key.

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

See also

- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“ULConnectionParms members” on page 123](#)
- ◆ [“ChangeEncryptionKey method” on page 105](#)

Password property

Specifies the password for the authenticated user.

Syntax

Visual Basic

Public Property **Password** As String

C#

```
public string Password {get;set;}
```

Property value

A string specifying a database user ID. The default is a null reference (Nothing in Visual Basic).

Remarks

Passwords are case sensitive.

When a database is created, the password for the DBA user ID is set to sql.

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

See also

- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“ULConnectionParms members” on page 123](#)
- ◆ [“UserID property” on page 131](#)

UserID property

Specifies an authenticated user for the database.

Syntax**Visual Basic**

Public Property **UserID** As String

C#

```
public string UserID {get;set;}
```

Property value

A string specifying a database user ID. The default value is a null reference (Nothing in Visual Basic).

Remarks

User IDs are case-insensitive.

Databases are initially created with a single authenticated user named DBA.

If both the user ID and password are not supplied, the user DBA with password sql are used. To make the database more secure, change the user DBA's password or create new users (using `ULConnection.GrantConnectTo`) and remove the DBA user (using `ULConnection.RevokeConnectFrom`).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

See also

- ◆ [“ULConnectionParms class” on page 122](#)
- ◆ [“ULConnectionParms members” on page 123](#)
- ◆ [“Password property” on page 130](#)
- ◆ [“GrantConnectTo method” on page 114](#)
- ◆ [“RevokeConnectFrom method” on page 116](#)

ToString method

Returns the string representation of this instance.

Syntax

Visual Basic

Overrides Public Function **ToString()** As String

C#

public override string **ToString()**;

Return value

The string representation of this instance as a semicolon-separated list keyword=value pairs.

UnusedEvent event

Unused.

Syntax

Visual Basic

Public Event **UnusedEvent** As ULConnectionParms.UnusedEventHandler

C#

public event ULConnectionParms.UnusedEventHandler **UnusedEvent**;

Remarks

This public Event is provided to fix a Visual Studio .NET bug relating to the integration of this class in Visual Basic .NET projects. It has no functional use.

ULConnectionParms.UnusedEventHandler delegate

UL Ext.: Unused.

Syntax

Visual Basic

```
Delegate Sub ULConnectionParms.UnusedEventHandler( _  
    ByVal sender As Object, _  
    ByVal args As System.EventArgs _  
)
```

C#

```
delegate void ULConnectionParms.UnusedEventHandler(  
    object sender,  
    System.EventArgs args  
);
```

Parameters

- ♦ **sender** Object that is the sender.
- ♦ **args** Event arguments.

Remarks

This public Delegate is provided to fix a Visual Studio .NET bug relating to the integration of this class in Visual Basic .NET projects. It has no functional use.

ULCreateParms class

UL Ext.: Builds a string of creation-time options for creating an UltraLite database.

Syntax

Visual Basic

Public Class **ULCreateParms**

C#

public class **ULCreateParms**

Remarks

A ULCreateParms object is used to specify the parameters for creating a database (ULDatabaseManager.CreateDatabase(string,byte[],string)).

Leading and trailing spaces are ignored in all string values. Values must not contain leading or trailing spaces, or a semicolon (;), or begin with either a single quote (') or a double quote (").

Once you have supplied all the creation parameters by setting the appropriate properties on a ULCreateParms object, you create a creation parameters string using the ULCreateParms.ToString. The resulting string can then be used as the createParms parameter of the ULDatabaseManager.CreateDatabase(string,byte[],string) method.

For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

See also

- ◆ [“ULCreateParms members” on page 134](#)
- ◆ [“GetDatabaseProperty method” on page 188](#)
- ◆ [“CreateDatabase method” on page 179](#)
- ◆ [“ToString method” on page 143](#)

ULCreateParms members

Public constructors

Member name	Description
ULCreateParms constructor	Initializes a ULCreateParms instance with its default values.

Public properties

Member name	Description
CaseSensitive property	Specifies whether the new database should be case sensitive when comparing string values.

Member name	Description
ChecksumLevel property	Specifies the level of database page checksums enabled for the new database.
DateFormat property	Specifies the date format used for string conversions by the new database.
DateOrder property	Specifies the date order used for string conversions by the new database.
FIPS property	Specifies whether the new database should be using AES_FIPS encryption or AES encryption.
MaxHashSize property	Specifies the default maximum number of bytes to use for index hashing in the new database.
NearestCentury property	Specifies the nearest century used for string conversions by the new database.
Obfuscate property	Specifies whether the new database should be using obfuscation (simple encryption) or not.
PageSize property	Specifies the page size of the new database, in bytes or kilobytes.
Precision property	Specifies the floating-point precision used for string conversions by the new database.
Scale property	Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the new database.
TimeFormat property	Specifies the time format used for string conversions by the new database.
TimestampFormat property	Specifies the timestamp format used for string conversions by the new database.
TimestampIncrement property	Specifies the minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).
UTF8Encoding property	Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Public methods

Member name	Description
ToString method	Returns the string representation of this instance.

ULCreateParms constructor

Initializes a ULCreateParms instance with its default values.

Syntax**Visual Basic**

```
Public Sub New()
```

C#

```
public ULCreateParms();
```

CaseSensitive property

Specifies whether the new database should be case sensitive when comparing string values.

Syntax**Visual Basic**

```
Public Property CaseSensitive As Boolean
```

C#

```
public bool CaseSensitive {get;set;}
```

Property value

True if the database should be case sensitive, false if the database should be case insensitive. The default is false.

Remarks

CaseSensitive only affects how string data is compared and sorted. Database identifiers such as table names, column names, index names, and connection user IDs are always case insensitive. Connection passwords and database encryption keys are always case sensitive.

ChecksumLevel property

Specifies the level of database page checksums enabled for the new database.

Syntax**Visual Basic**

```
Public Property ChecksumLevel As Integer
```

C#

```
public int ChecksumLevel {get;set;}
```

Property value

An integer specifying the checksum level. Valid values are 0, 1, and 2. The default is 0.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

DateFormat property

Specifies the date format used for string conversions by the new database.

Syntax

Visual Basic

Public Property **DateFormat** As String

C#

public string **DateFormat** {get;set;}

Property value

A string specifying the date format. If the value is a null reference (Nothing in Visual Basic), the database will use "YYYY-MM-DD". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

DateOrder property

Specifies the date order used for string conversions by the new database.

Syntax

Visual Basic

Public Property **DateOrder** As UDateOrder

C#

public UDateOrder **DateOrder** {get;set;}

Property value

A UDateOrder value identifying the date order for string conversions. The default is YMD.

See also

- ◆ [“ULCreateParms class” on page 134](#)
- ◆ [“ULCreateParms members” on page 134](#)
- ◆ [“UDateOrder enumeration” on page 237](#)

FIPS property

Specifies whether the new database should be using AES_FIPS encryption or AES encryption.

Syntax

Visual Basic

Public Property **FIPS** As Boolean

C#

```
public bool FIPS {get;set;}
```

Property value

True if the database should be encrypted using AES_FIPS, false if the database should be encrypted with AES. The default is false.

Remarks

Encryption must be turned on by supplying a value for the connection parameter EncryptionKey when the new database is created. If FIPS is set true and no encryption key is supplied, the ULDatabaseManager.CreateDatabase(string,byte[],string) method will fail with a missing encryption key error.

See also

- ◆ [“ULCreateParms class” on page 134](#)
- ◆ [“ULCreateParms members” on page 134](#)
- ◆ [“EncryptionKey property” on page 130](#)
- ◆ [“CreateDatabase method” on page 179](#)

MaxHashSize property

Specifies the default maximum number of bytes to use for index hashing in the new database.

Syntax

Visual Basic

Public Property **MaxHashSize** As Integer

C#

```
public int MaxHashSize {get;set;}
```

Property value

An integer specifying the maximum hash size. The value must be in the range [0,32]. The default is 8.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

NearestCentury property

Specifies the nearest century used for string conversions by the new database.

Syntax

Visual Basic

Public Property **NearestCentury** As Integer

C#

```
public int NearestCentury {get;set;}
```

Property value

An integer specifying the nearest century. The value must be in the range [0,100]. The default is 50.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

Obfuscate property

Specifies whether the new database should be using obfuscation (simple encryption) or not.

Syntax

Visual Basic

Public Property **Obfuscate** As Boolean

C#

```
public bool Obfuscate {get;set;}
```

Property value

True if the database should be encrypted using obfuscation, false if the database should not be obfuscated. The default is false.

Remarks

This option is ignored if FIPS encryption is turned on (ULCreateParms.FIPS). If obfuscation is turned on and a value is supplied for the connection parameter EncryptionKey (DBKEY) when the new database is created, the encryption key will be ignored.

See also

- ◆ [“ULCreateParms class” on page 134](#)
- ◆ [“ULCreateParms members” on page 134](#)
- ◆ [“FIPS property” on page 138](#)

PageSize property

Specifies the page size of the new database, in bytes or kilobytes.

Syntax

Visual Basic

Public Property **PageSize** As Integer

C#

```
public int PageSize {get;set;}
```

Property value

An integer specifying the page size in bytes. Valid values are 1024 (1K), 2048 (2K), 4096 (4K), 8192 (8K), 16384 (16K). The default is 4096.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

Precision property

Specifies the floating-point precision used for string conversions by the new database.

Syntax

Visual Basic

Public Property **Precision** As Integer

C#

```
public int Precision {get;set;}
```

Property value

An integer specifying the precision. The value must be in the range [1,127]. The default is 30.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

See also

- ◆ [“ULCreateParms class” on page 134](#)
- ◆ [“ULCreateParms members” on page 134](#)
- ◆ [“Scale property” on page 141](#)

Scale property

Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the new database.

Syntax

Visual Basic

Public Property **Scale** As Integer

C#

public int **Scale** {get;set;}

Property value

An integer specifying the scale. The value must be in the range [0,127]. The default is 6.

Remarks

Scale must be less than or equal to the Precision. If Scale is greater than the Precision, an error will occur while creating the database.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

TimeFormat property

Specifies the time format used for string conversions by the new database.

Syntax

Visual Basic

Public Property **TimeFormat** As String

C#

public string **TimeFormat** {get;set;}

Property value

A string specifying the time format. If the value is a null reference (Nothing in Visual Basic), the database will use "HH:NN:SS.SSS". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

TimestampFormat property

Specifies the timestamp format used for string conversions by the new database.

Syntax

Visual Basic

Public Property **TimestampFormat** As String

C#

public string **TimestampFormat** {get;set;}

Property value

A string specifying the timestamp format. If the value is a null reference (Nothing in Visual Basic), the database will use "YYYY-MM-DD HH:NN:SS.SSS". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

Exceptions

- ◆ [ArgumentException](#) - The value contained a semicolon (;), or began with either a single quote (') or a double quote (").

TimestampIncrement property

Specifies the minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).

Syntax

Visual Basic

Public Property **TimestampIncrement** As Integer

C#

public int **TimestampIncrement** {get;set;}

Property value

An integer specifying the timestamp increment. The value must be in the range [1,60000000]. The default is 1.

Exceptions

- ◆ [ArgumentException](#) - The value is invalid.

UTF8Encoding property

Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Syntax**Visual Basic**

Public Property **UTF8Encoding** As Boolean

C#

```
public bool UTF8Encoding {get;set;}
```

Property value

True if the database should use the UTF8 character set, false if the database should use the character set associated with the collation. The default is false.

Remarks

Choose to use the UTF8 character set if you wish to store characters that are not in the character set associated with the collation. For example, you create a database with the 1252LATIN1 collation because you want US sorting but specify UTF8Encoding true because you want to store international addresses as they are spelled locally.

For databases used on Symbian OS devices, you must set UTF8Encoding true.

For databases used on Palm OS devices, you must leave UTF8Encoding false.

ToString method

Returns the string representation of this instance.

Syntax**Visual Basic**

Overrides Public Function **ToString()** As String

C#

```
public override string ToString();
```

Return value

The string representation of this instance as a semicolon-separated list keyword=value pairs.

ULCursorSchema class

UL Ext.: Represents the schema of an UltraLite.NET cursor.

Syntax

Visual Basic

MustInherit Public Class **ULCursorSchema**

C#

public abstract class **ULCursorSchema**

Remarks

This class is an abstract base class of the ULTableSchema class and the ULResultSetSchema class.

Note to users porting from the iAnywhere.UltraLite namespace: Column IDs are 0-based, not 1-based as they are in the iAnywhere.UltraLite namespace.

See also

- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULResultSetSchema class” on page 342](#)

ULCursorSchema members

Public properties

Member name	Description
ColumnCount property	Returns the number of columns in the cursor.
IsOpen property	Checks whether the cursor schema is currently open.
Name property	Returns the name of the cursor.

Public methods

Member name	Description
GetColumnID method	Returns the column ID of the named column.
GetColumnName method	Returns the name of the column identified by the specified column ID.
GetColumnPrecision method	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnScale method	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Member name	Description
GetColumnSize method	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnSQLName method	Returns the name of the column identified by the specified column ID.
GetColumnULDbType method	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetSchemaTable method	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

Protected methods

Member name	Description
Finalize method	Releases the unmanaged resources used by the ULCursorSchema.

ColumnCount property

Returns the number of columns in the cursor.

Syntax

Visual Basic

Public Readonly Property **ColumnCount** As Short

C#

```
public short ColumnCount {get;}
```

Property value

The number of columns in the cursor or 0 if the cursor schema is closed.

Remarks

Column IDs range from 0 to ColumnCount-1, inclusive.

Column IDs and count might change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

IsOpen property

Checks whether the cursor schema is currently open.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

public bool **IsOpen** {get;}

Property value

True if the cursor schema is currently open, false if the cursor schema is closed.

Name property

Returns the name of the cursor.

Syntax

Visual Basic

Public Readonly Property **Name** As String

C#

public string **Name** {get;}

Property value

The name of the cursor as a string.

Finalize method

Releases the unmanaged resources used by the ULCursorSchema.

Syntax

Visual Basic

Overrides Protected Sub **Finalize()**

C#

protected override void **Finalize()**;

GetColumnID method

Returns the column ID of the named column.

Syntax

Visual Basic

```
Public Function GetColumnID( _  
    ByVal name As String _  
) As Short
```

C#

```
public short GetColumnID(  
    string name  
);
```

Parameters

- ◆ **name** The name of the column.

Return value

The column ID of the named column.

Remarks

Column IDs range from 0 to ColumnCount-1, inclusive.

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and counts might change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnName method

Returns the name of the column identified by the specified column ID.

Syntax

Visual Basic

```
Public Function GetColumnName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and count may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnPrecision method

Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Syntax**Visual Basic**

```
Public Function GetColumnPrecision( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnPrecision(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The precision of the specified numeric column.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnScale method

Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Syntax**Visual Basic**

```
Public Function GetColumnScale( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnScale(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The scale of the specified numeric column.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnSize method

Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).

Syntax

Visual Basic

```
Public Function GetColumnSize( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnSize(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The size of the specified sized column.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnSQLName method

Returns the name of the column identified by the specified column ID.

Syntax

Visual Basic

```
Public Function GetColumnSQLName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnSQLName(  
    int columnID  
);
```


Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are using aliases, the name of the column is the alias.

The GetColumnSQLName method differs from the GetColumnName in that for non-aliased, non-computed columns GetColumnSQLName always returns just the name of the column (without the table name as a prefix). While this behavior more closely resembles the behavior of other ADO.NET providers, it is more likely to produce non-unique names.

Column IDs and count may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“GetColumnName method” on page 147](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnULDbType method

Returns the UltraLite.NET data type of the column identified by the specified column ID.

Syntax

Visual Basic

```
Public Function GetColumnULDbType( _  
    ByVal columnID As Integer _  
) As ULDbType
```

C#

```
public ULDbType GetColumnULDbType(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

A ULDbType enumerated integer.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ULDbType enumeration” on page 238](#)

GetSchemaTable method

Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

Syntax

Visual Basic

Public Function **GetSchemaTable()** As DataTable

C#

public DataTable **GetSchemaTable();**

Return value

A System.Data.DataTable that describes the column schema.

Remarks

For more information, see [GetSchemaTable method](#).

See also

- ◆ [“ULCursorSchema class” on page 144](#)
- ◆ [“ULCursorSchema members” on page 144](#)
- ◆ [DataTable](#)
- ◆ [“ULDataReader class” on page 195](#)

ULDataAdapter class

Represents a set of commands and a database connection used to fill a System.Data.DataSet and to update a database.

Syntax

Visual Basic

NotInheritable Public Class **ULDataAdapter**
Inherits Component

C#

public sealed class **ULDataAdapter** :
Component

Remarks

The System.Data.DataSet provides a way to work with data offline; that is, away from your UltraLite database. The ULDataAdapter provides methods to associate a System.Data.DataSet with a set of SQL statements.

Since UltraLite is a local database and MobiLink has conflict resolution, the use of the ULDataAdapter is limited. For most purposes, the ULDataReader or the ULTable provide more efficient access to data.

Implements: System.Data.IDbDataAdapter, System.Data.IDataAdapter, System.IDisposable

See also

- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [IDbDataAdapter](#)
- ◆ [IDataAdapter](#)
- ◆ [IDisposable](#)

ULDataAdapter members

Public constructors

Member name	Description
ULDataAdapter constructor	Initializes a ULDataAdapter object.
ULDataAdapter constructor	Initializes a ULDataAdapter object with the specified SELECT statement.
ULDataAdapter constructor	Initializes a ULDataAdapter object with the specified SELECT statement and connection.

Member name	Description
ULDataAdapter constructor	Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

Public properties

Member name	Description
AcceptChangesDuringFill property	Specifies whether System.Data.DataRow.AcceptChanges is called on a System.Data.DataRow after it is added to the System.Data.DataTable class.
ContinueUpdateOnError property	Specifies whether to generate an exception when an error is encountered during a row update.
DeleteCommand property	Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update(System.Data.DataSet) is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.
InsertCommand property	Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update(System.Data.DataSet) is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.
MissingMappingAction property	Determines the action to take when incoming data does not have a matching table or column.
MissingSchemaAction property	Determines the action to take when the existing System.Data.DataSet schema does not match incoming data.
SelectCommand property	Specifies a ULCommand that is used during ULDataAdapter.Fill (System.Data.DataSet) or ULDataAdapter.FillSchema (System.Data.DataSet, System.Data.SchemaType) to obtain a result set from the database for copying into a System.Data.DataSet.
TableMappings property	Returns a collection that provides the master mapping between a source table and a System.Data.DataTable
UpdateCommand property	Specifies a ULCommand object that is executed against the database when ULDataAdapter.Update(System.Data.DataSet) is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Public methods

Member name	Description
Fill method	
Fill method	Adds or refreshes rows into the named System.Data.DataTable of the specified System.Data.DataSet. The System.Data.DataTable named is created and added to the System.Data.DataSet if necessary.

Member name	Description
Fill method	Adds or refreshes the specified range of rows into the named System.Data.DataTable of the specified System.Data.DataSet. The System.Data.DataTable named is created and added to the System.Data.DataSet if necessary.
Fill method	Adds or refreshes rows into the specified System.Data.DataTable.
FillSchema method	Adds a System.Data.DataTable named "Table" to a System.Data.DataSet and configures the schema to match the schema in the data source.
FillSchema method	Adds a System.Data.DataTable named "Table" to a System.Data.DataSet and configures the schema to match the schema in the data source.
FillSchema method	Configures the schema of a System.Data.DataTable to match the schema in the data source.
GetFillParameters method	Returns the parameters set by the user when executing a SELECT statement.
Update method	Updates the rows in the database with the changes made to the System.Data.DataTable named "Table" of the specified System.Data.DataSet.
Update method	Updates the rows in the database with the changes made to the named System.Data.DataTable of the specified System.Data.DataSet.
Update method	Updates the rows in the database with the changes made to the specified System.Data.DataTable.
Update method	Updates the rows in the database with the changes made to the specified System.Data.DataRow array.

Public events

Member name	Description
FillError event	Occurs when an error is detected during a fill operation.
RowUpdated event	Occurs during an update after a command is executed against the data source. When an attempt to update is made, the event fires.
RowUpdating event	Occurs during an update before a command is executed against the data source. When an attempt to update is made, the event fires.

ULDataAdapter constructor

Initializes a ULDataAdapter object.

Syntax

Visual Basic

Overloads Public Sub **New()**

C#

public **ULDataAdapter()**;

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULDataAdapter constructor” on page 156](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)

ULDataAdapter constructor

Initializes a ULDataAdapter object with the specified SELECT statement.

Syntax

Visual Basic

Overloads Public Sub **New**(
 ByVal *selectCommand* As ULCommand
)

C#

public **ULDataAdapter**(
 ULCommand *selectCommand*
);

Parameters

- ◆ **selectCommand** A ULCommand object that is used during System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet) to select records from the data source for placement in the System.Data.DataSet.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULDataAdapter constructor” on page 155](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DbDataAdapter.Fill](#)
- ◆ [DataSet](#)

ULDataAdapter constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal selectCommandText As String, _  
    ByVal selectConnection As ULConnection _  
)
```

C#

```
public ULDataAdapter(  
    string selectCommandText,  
    ULConnection selectConnection  
);
```

Parameters

- ◆ **selectCommandText** A SELECT statement to be used by the ULDataAdapter.SelectCommand of the ULDataAdapter.
- ◆ **selectConnection** A ULConnection object that defines a connection to a database.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULDataAdapter constructor” on page 155](#)
- ◆ [“ULDataAdapter constructor” on page 156](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“ULConnection class” on page 90](#)

ULDataAdapter constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal selectCommandText As String, _  
    ByVal selectConnectionString As String _  
)
```

C#

```
public ULDataAdapter(  
    string selectCommandText,
```

```
        string selectConnectionString  
    );
```

Parameters

- ◆ **selectCommandText** A SELECT statement to be used by the ULDataAdapter.SelectCommand of the ULDataAdapter.
- ◆ **selectConnectionString** A connection string for an UltraLite.NET database.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULDataAdapter constructor” on page 155](#)
- ◆ [“ULDataAdapter constructor” on page 156](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“SelectCommand property” on page 162](#)

AcceptChangesDuringFill property

Specifies whether System.Data.DataRow.AcceptChanges is called on a System.Data.DataRow after it is added to the System.Data.DataTable class.

Syntax

Visual Basic

Public Property **AcceptChangesDuringFill** As Boolean

C#

```
public bool AcceptChangesDuringFill {get;set;}
```

Property value

True to specify that the ULDataAdapter should call the System.Data.DataRow.AcceptChanges function on the System.Data.DataRow class; false if AcceptChanges is not to be called, and the newly added rows are treated as inserted rows. The default is true.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataRow.AcceptChanges](#)
- ◆ [DataRow](#)
- ◆ [DataTable](#)

ContinueUpdateOnError property

Specifies whether to generate an exception when an error is encountered during a row update.

Syntax**Visual Basic**

Public Property **ContinueUpdateOnError** As Boolean

C#

public bool **ContinueUpdateOnError** {get;set;}

Property value

True to continue the update without generating an exception, false to generate an exception. The default is false.

Remarks

If ContinueUpdateOnError is true, no exception is thrown when an error occurs during the update of a row. The update of the row is skipped and the error information is placed in the System.Data.DataRow.RowError method of the row. The ULDataAdapter continues to update subsequent rows.

If ContinueUpdateOnError is false, an exception is thrown when an error occurs.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataRow.RowError](#)

DeleteCommand property

Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update (System.Data.DataSet) is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

Syntax**Visual Basic**

Public Property **DeleteCommand** As ULCommand

C#

public ULCommand **DeleteCommand** {get;set;}

Property value

A ULCommand object that is executed to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

Remarks

When DeleteCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The DeleteCommand maintains a reference to the existing ULCommand.

This is the strongly-typed version of System.Data.IDbDataAdapter.DeleteCommand .

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DataSet](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [IDbDataAdapter.DeleteCommand](#)

InsertCommand property

Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update (System.Data.DataSet) is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

Syntax**Visual Basic**

Public Property **InsertCommand** As ULCommand

C#

```
public ULCommand InsertCommand {get;set;}
```

Property value

A ULCommand object that is executed to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

Remarks

When InsertCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The InsertCommand maintains a reference to the existing ULCommand.

This is the strongly-typed version of System.Data.IDbDataAdapter.InsertCommand .

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DataSet](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [IDbDataAdapter.InsertCommand](#)

MissingMappingAction property

Determines the action to take when incoming data does not have a matching table or column.

Syntax

Visual Basic

NotOverridable Public Property **MissingMappingAction** As MissingMappingAction _
Implements IDataAdapter.MissingMappingAction

C#

public MissingMappingAction **MissingMappingAction** {get;set;}

Property value

One of the System.Data.MissingMappingAction values. The default is System.Data.MissingMappingAction.Passthrough.

Implements

[IDataAdapter.MissingMappingAction](#)

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [MissingMappingAction](#)
- ◆ [MissingMappingAction.Passthrough](#)

MissingSchemaAction property

Determines the action to take when the existing System.Data.DataSet schema does not match incoming data.

Syntax

Visual Basic

NotOverridable Public Property **MissingSchemaAction** As MissingSchemaAction _
Implements IDataAdapter.MissingSchemaAction

C#

public MissingSchemaAction **MissingSchemaAction** {get;set;}

Property value

One of the System.Data.MissingSchemaAction values. The default is System.Data.MissingSchemaAction.Add.

Implements

[IDataAdapter.MissingSchemaAction](#)

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataSet](#)

- ◆ [MissingSchemaAction](#)
- ◆ [MissingSchemaAction.Add](#)

SelectCommand property

Specifies a ULCommand that is used during ULDataAdapter.Fill(System.Data.DataSet) or ULDataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType) to obtain a result set from the database for copying into a System.Data.DataSet.

Syntax

Visual Basic

Public Property **SelectCommand** As ULCommand

C#

public ULCommand **SelectCommand** {get;set;}

Property value

A ULCommand object that is executed to fill the System.Data.DataSet.

Remarks

When SelectCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The SelectCommand maintains a reference to the existing ULCommand.

If the SelectCommand does not return any rows, no tables are added to the System.Data.DataSet, and no exception is raised. The SELECT statement can also be specified in the ULDataAdapter(ULCommand), ULDataAdapter(String, ULConnection), or ULDataAdapter(String, String) constructors.

This is the strongly-typed version of System.Data.IDbDataAdapter.SelectCommand .

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DataSet](#)
- ◆ [“Fill method” on page 164](#)
- ◆ [“FillSchema method” on page 168](#)
- ◆ [“ULDataAdapter constructor” on page 156](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [“ULDataAdapter constructor” on page 157](#)
- ◆ [IDbDataAdapter.SelectCommand](#)

TableMappings property

Returns a collection that provides the master mapping between a source table and a System.Data.DataTable

Syntax

Visual Basic

Public Readonly Property **TableMappings** As DataTableMappingCollection

C#

public DataTableMappingCollection **TableMappings** {get;}

Property value

A collection of System.Data.Common.DataTableMapping objects providing the master mapping between source tables and System.Data.DataTables. The default value is an empty collection.

Remarks

When reconciling changes, the ULDataAdapter uses the System.Data.Common.DataTableMappingCollection collection to associate the column names used by the data source with the column names used by the System.Data.DataSet.

This is the strongly-typed version of System.Data.IDataAdapter.TableMappings.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataTableMapping](#)
- ◆ [DataTable](#)
- ◆ [DataTableMappingCollection](#)
- ◆ [DataSet](#)
- ◆ [IDataAdapter.TableMappings](#)

UpdateCommand property

Specifies a ULCommand object that is executed against the database when ULDataAdapter.Update (System.Data.DataSet) is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Syntax

Visual Basic

Public Property **UpdateCommand** As ULCommand

C#

public ULCommand **UpdateCommand** {get;set;}

Property value

A ULCommand object that is executed to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Remarks

When UpdateCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The UpdateCommand maintains a reference to the existing ULCommand.

If execution of this command returns rows, these rows may be merged with the System.Data.DataSet depending on how you set the ULCommand.UpdatedRowSource of the ULCommand object.

This is the strongly-typed version of System.Data.IDbDataAdapter.UpdateCommand .

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DataSet](#)
- ◆ [“Update method” on page 172](#)
- ◆ [“UpdatedRowSource property” on page 69](#)
- ◆ [IDbDataAdapter.UpdateCommand](#)

Fill method

Adds or refreshes rows into the named System.Data.DataTable of the specified System.Data.DataSet. The System.Data.DataTable named is created and added to the System.Data.DataSet if neccessary.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function Fill( _  
    ByVal dataSet As DataSet _  
) As Integer _  
    Implements IDDataAdapter.Fill
```

C#

```
public int Fill(  
    DataSet dataSet  
);
```

Parameters

- ◆ **dataSet**

Return value

Implements

[IDDataAdapter.Fill](#)

Fill method

Adds or refreshes rows into the named System.Data.DataTable of the specified System.Data.DataSet. The System.Data.DataTable named is created and added to the System.Data.DataSet if neccessary.

Syntax

Visual Basic

```
Overloads Public Function Fill( _  
    ByVal dataSet As DataSet, _  
    ByVal srcTable As String _  
) As Integer
```

C#

```
public int Fill(  
    DataSet dataSet,  
    string srcTable  
);
```

Parameters

- ◆ **dataSet** A System.Data.DataSet to fill with records and optionally schema.
- ◆ **srcTable** The name of the source table to use for table mapping.

Return value

The number of rows successfully added or refreshed in the System.Data.DataSet.

Remarks

For large result sets, this can have a significant performance impact. An alternative is to use an ULDataReader when a read-only result set is sufficient, perhaps with SQL statements (ExecuteNonQuery) to carry out modifications. Another alternative is to use a ULTable that allows read-write access to the database.

If ULDataAdapter.SelectCommand does not return any rows, no tables are added to the System.Data.DataSet and no exception is raised.

For more information, see [IDataAdapter.Fill](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataSet* or *srcTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error or, the System.Data.DataTable is missing from the System.Data.DataSet and ULDataAdapter.MissingSchemaAction is set to System.Data.MissingSchemaAction.Error.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)

- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [IDataAdapter.Fill](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)
- ◆ [“MissingSchemaAction property” on page 161](#)
- ◆ [MissingSchemaAction.Error](#)

Fill method

Adds or refreshes the specified range of rows into the named System.Data.DataTable of the specified System.Data.DataSet. The System.Data.DataTable named is created and added to the System.Data.DataSet if neccessary.

Syntax

Visual Basic

```
Overloads Public Function Fill( _  
    ByVal dataSet As DataSet, _  
    ByVal startRecord As Integer, _  
    ByVal maxRecords As Integer, _  
    ByVal srcTable As String _  
    ) As Integer
```

C#

```
public int Fill(  
    DataSet dataSet,  
    int startRecord,  
    int maxRecords,  
    string srcTable  
);
```

Parameters

- ◆ **dataSet** A System.Data.DataSet to fill with records and optionally schema.
- ◆ **startRecord** The zero-based record number with which to start.
- ◆ **maxRecords** The maximum number of records to be read into the System.Data.DataSet. A value of 0 gets all records found after the start record.
- ◆ **srcTable** The name of the source table to use for table mapping.

Return value

The number of rows successfully added or refreshed in the System.Data.DataSet.

Remarks

Even if you use the `startRecord` argument to limit the number of records that are copied to the `System.Data.DataSet`, all records in the `ULDataAdapter` query are fetched from the database to the client. For large result sets, this can have a significant performance impact. An alternative is to use an `ULDataReader` when a read-only result set is sufficient, perhaps with SQL statements (`ExecuteNonQuery`) to carry out modifications. Another alternative is to use a `ULTable` that allows read-write access to the database.

If `ULDataAdapter.SelectCommand` does not return any rows, no tables are added to the `System.Data.DataSet` and no exception is raised.

For more information, see [IDataAdapter.Fill](#).

Exceptions

- ◆ [ArgumentException](#) - The *startRecord* or *maxRecords* parameter was less than zero.
- ◆ [System.ArgumentNullException](#) - The *dataSet* or *srcTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The `ULDataAdapter.SelectCommand` is invalid or, the table mapping is missing and `ULDataAdapter.MissingMappingAction` is set to `System.Data.MissingMappingAction.Error` or, the `System.Data.DataTable` is missing from the `System.Data.DataSet` and `ULDataAdapter.MissingSchemaAction` is set to `System.Data.MissingSchemaAction.Error`.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [IDataAdapter.Fill](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)
- ◆ [“MissingSchemaAction property” on page 161](#)
- ◆ [MissingSchemaAction.Error](#)

Fill method

Adds or refreshes rows into the specified `System.Data.DataTable`.

Syntax

Visual Basic

```
Overloads Public Function Fill( _  
    ByVal dataTable As DataTable _  
) As Integer
```

C#

```
public int Fill(  
    DataTable dataTable  
);
```

Parameters

- ◆ **dataTable** A System.Data.DataTable to fill with records and optionally schema.

Return value

The number of rows successfully added or refreshed in the System.Data.DataTable.

Remarks

For large result sets, this can have a significant performance impact. An alternative is to use an ULDataReader when a read-only result set is sufficient, perhaps with SQL statements (ExecuteNonQuery) to carry out modifications. Another alternative is to use a ULTable that allows read-write access to the database.

If ULDataAdapter.SelectCommand does not return any rows, no tables are added and no exception is raised.

For more information, see [IDataAdapter.Fill](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [IDataAdapter.Fill](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

FillSchema method

Adds a System.Data.DataTable named "Table" to a System.Data.DataSet and configures the schema to match the schema in the data source.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function FillSchema( _  
    ByVal dataSet As DataSet, _  
    ByVal schemaType As SchemaType _  
) As DataTable() _  
    Implements IDDataAdapter.FillSchema
```

C#

```
public DataTable[] FillSchema(  
    DataSet dataSet,  
    SchemaType schemaType  
);
```

Parameters

- ◆ **dataSet** A System.Data.DataSet to fill with the schema.
- ◆ **schemaType** One of the System.Data.SchemaType values that specify how to insert the schema.

Return value

A reference to a collection of System.Data.DataTable objects that were added to the System.Data.DataSet.

Remarks

For more information, see [IDDataAdapter.FillSchema](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataSet* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error.

Implements

[IDDataAdapter.FillSchema](#)

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [IDDataAdapter.Fill](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

FillSchema method

Adds a System.Data.DataTable named "Table" to a System.Data.DataSet and configures the schema to match the schema in the data source.

Syntax

Visual Basic

```
Overloads Public Function FillSchema( _  
    ByVal dataSet As DataSet, _  
    ByVal schemaType As SchemaType, _  
    ByVal srcTable As String _  
) As DataTable()
```

C#

```
public DataTable[] FillSchema(  
    DataSet dataSet,  
    SchemaType schemaType,  
    string srcTable  
);
```

Parameters

- ◆ **dataSet** A System.Data.DataSet to fill with the schema.
- ◆ **schemaType** One of the System.Data.SchemaType values that specify how to insert the schema.
- ◆ **srcTable** The name of the source table to use for table mapping.

Return value

A reference to a collection of System.Data.DataTable objects that were added to the System.Data.DataSet.

Remarks

For more information, see [DbDataAdapter.FillSchema](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataSet* or *srcTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataSet](#)
- ◆ [SchemaType](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULTable class” on page 419](#)

- ◆ [“SelectCommand property” on page 162](#)
- ◆ [IDataAdapter.Fill](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

FillSchema method

Configures the schema of a System.Data.DataTable to match the schema in the data source.

Syntax

Visual Basic

```
Overloads Public Function FillSchema( _  
    ByVal dataTable As DataTable, _  
    ByVal schemaType As SchemaType _  
) As DataTable
```

C#

```
public DataTable FillSchema(  
    DataTable dataTable,  
    SchemaType schemaType  
);
```

Parameters

- ◆ **dataTable** A System.Data.DataTable to fill with the schema.
- ◆ **schemaType** One of the System.Data.SchemaType values that specify how to insert the schema.

Return value

A reference to the System.Data.DataTable object that contains the schema.

Remarks

For more information, see [DbDataAdapter.FillSchema](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [SchemaType](#)
- ◆ [“SelectCommand property” on page 162](#)

GetFillParameters method

Returns the parameters set by the user when executing a SELECT statement.

Syntax

Visual Basic

```
Public Function GetFillParameters() As ULParameter()
```

C#

```
public ULParameter[] GetFillParameters();
```

Return value

An array of ULParameter objects that contains the parameters set by the user.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULParameter class” on page 277](#)

Update method

Updates the rows in the database with the changes made to the System.Data.DataTable named "Table" of the specified System.Data.DataSet.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function Update( _  
    ByVal dataSet As DataSet _  
) As Integer _  
    Implements IDDataAdapter.Update
```

C#

```
public int Update(  
    DataSet dataSet  
);
```

Parameters

- ◆ **dataSet** A System.Data.DataSet to update with records from.

Return value

The number of rows successfully updated from the System.Data.DataSet.

Remarks

For more information, see [IDataAdapter.Update](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataSet* parameter is invalid.
- ◆ [InvalidOperationException](#) - The `ULDataAdapter.SelectCommand` is invalid or, the table mapping is missing and `ULDataAdapter.MissingMappingAction` is set to `System.Data.MissingMappingAction.Error`, or the `System.Data.DataTable` is missing from the `System.Data.DataSet`.

Implements

[IDataAdapter.Update](#)

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DataSet](#)
- ◆ [IDataAdapter.Update](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

Update method

Updates the rows in the database with the changes made to the named `System.Data.DataTable` of the specified `System.Data.DataSet`.

Syntax

Visual Basic

```
Overloads Public Function Update( _  
    ByVal dataSet As DataSet, _  
    ByVal srcTable As String _  
) As Integer
```

C#

```
public int Update(  
    DataSet dataSet,  
    string srcTable  
);
```

Parameters

- ◆ **dataSet** A `System.Data.DataSet` to update with records from.
- ◆ **srcTable** The name of the source table to use for table mapping.

Return value

The number of rows successfully updated from the `System.Data.DataSet`.

Remarks

For more information, see [DbDataAdapter.Update](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataSet* or *srcTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The *ULDataAdapter.SelectCommand* is invalid or, the table mapping is missing and *ULDataAdapter.MissingMappingAction* is set to *System.Data.MissingMappingAction.Error* or, the *System.Data.DataTable* is missing from the *System.Data.DataSet*.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DataSet](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

Update method

Updates the rows in the database with the changes made to the specified *System.Data.DataTable*.

Syntax**Visual Basic**

```
Overloads Public Function Update( _  
    ByVal dataTable As DataTable _  
) As Integer
```

C#

```
public int Update(  
    DataTable dataTable  
);
```

Parameters

- ◆ **dataTable** A *System.Data.DataTable* to update from.

Return value

The number of rows successfully updated from the *System.Data.DataTable*.

Remarks

For more information, see [DbDataAdapter.Update](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataTable* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataTable](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

Update method

Updates the rows in the database with the changes made to the specified System.Data.DataRow array.

Syntax

Visual Basic

```
Overloads Public Function Update( _  
    ByVal dataRows As DataRow() _  
) As Integer
```

C#

```
public int Update(  
    DataRow[] dataRows  
);
```

Parameters

- ◆ **dataRows** An array of System.Data.DataRow to update from.

Return value

The number of rows successfully updated from the System.Data.DataRow array.

Remarks

For more information, see [DbDataAdapter.Update](#).

Exceptions

- ◆ [System.ArgumentNullException](#) - The *dataRows* parameter is invalid.
- ◆ [InvalidOperationException](#) - The ULDataAdapter.SelectCommand is invalid or, the table mapping is missing and ULDataAdapter.MissingMappingAction is set to System.Data.MissingMappingAction.Error.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [DataRow](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [“SelectCommand property” on page 162](#)
- ◆ [“MissingMappingAction property” on page 160](#)
- ◆ [MissingMappingAction.Error](#)

FillError event

Occurs when an error is detected during a fill operation.

Syntax**Visual Basic**

Public Event **FillError** As FillErrorHandler

C#

public event FillErrorHandler **FillError**;

Remarks

The FillError event allows you to determine whether the fill operation should continue after the error occurs.

Examples of when the FillError event might occur are:

- ◆ The data being added to a DataSet cannot be converted to a common language runtime type without losing precision.
- ◆ The row being added contains data that violates a Constraint that must be enforced on a DataColumn in the DataSet.

To process row fill error events, you must create a System.Data.FillErrorHandler delegate and attach it to this event.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [FillErrorHandler](#)

RowUpdated event

Occurs during an update after a command is executed against the data source. When an attempt to update is made, the event fires.

Syntax**Visual Basic**

Public Event **RowUpdated** As ULRowUpdatedEventHandler

C#

public event ULRowUpdatedEventHandler **RowUpdated**;

Remarks

To process row updated events, you must create a ULRowUpdatedEventHandler delegate and attach it to this event.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULRowUpdatedEventHandler delegate” on page 348](#)

RowUpdating event

Occurs during an update before a command is executed against the data source. When an attempt to update is made, the event fires.

Syntax**Visual Basic**

Public Event **RowUpdating** As ULRowUpdatingEventHandler

C#

public event ULRowUpdatingEventHandler **RowUpdating**;

Remarks

To process row updating events, you must create a ULRowUpdatingEventHandler delegate and attach it to this event.

See also

- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataAdapter members” on page 153](#)
- ◆ [“ULRowUpdatedEventHandler delegate” on page 348](#)

ULDatabaseManager class

UL Ext.: Manages synchronization listeners and the UltraLite.NET runtime type. The ULDatabaseManager class also allows you to drop (delete) UltraLite.NET databases.

Syntax

Visual Basic

NotInheritable Public Class **ULDatabaseManager**

C#

public sealed class **ULDatabaseManager**

Remarks

This class is a singleton class whose only instance is accessible through the static (Shared in Visual Basic) `ULConnection.DatabaseManager`.

To use the UltraLite Engine runtime of UltraLite.NET, set `ULDatabaseManager.RuntimeType` to the appropriate value before using any other UltraLite.NET API.

See also

- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“DatabaseManager property” on page 98](#)
- ◆ [“RuntimeType property” on page 179](#)

ULDatabaseManager members

Public static properties (shared)

Member name	Description
RuntimeType property	Specifies the UltraLite.NET runtime type. The runtime type must be selected before using any other UltraLite.NET API.

Public methods

Member name	Description
CreateDatabase method	Creates a new UltraLite database.
DropDatabase method	Deletes the specified database. You cannot drop a database that has open connections.
SetActiveSyncListener method	Specifies the listener object used to process ActiveSync calls from the MobiLink provider for ActiveSync.
SetServerSyncListener method	Specifies the listener object used to process the specified server synchronization message.

Member name	Description
SignalSyncIsComplete method	Signals the MobiLink provider for ActiveSync that an application has completed synchronization.

RuntimeType property

Specifies the UltraLite.NET runtime type. The runtime type must be selected before using any other UltraLite.NET API.

Syntax

Visual Basic

Public Shared Property **RuntimeType** As ULRuntimeType

C#

```
public const ULRuntimeType RuntimeType {get;set;}
```

Property value

A ULRuntimeType value identifying the type of the unmanaged UltraLite.NET runtime.

Example

The following example selects the UltraLite Engine runtime and creates a connection.

```
' Visual Basic
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT
Dim conn As ULConnection = new ULConnection
' The RuntimeType is now locked

// C#
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT;
ULConnection conn = new ULConnection();
// The RuntimeType is now locked
```

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“ULRuntimeType enumeration” on page 353](#)

CreateDatabase method

Creates a new UltraLite database.

Syntax

Visual Basic

```
Public Sub CreateDatabase( _  
    ByVal connString As String, _
```

```
ByVal collationData As Byte(), _  
ByVal createParms As String _  
)
```

C#

```
public void CreateDatabase(  
    string connString,  
    byte[] collationData,  
    string createParms  
);
```

Parameters

- ♦ **connString** The parameters for identifying a database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [ULConnectionParms class](#).
- ♦ **collationData** The collation data specifying how the database will store and compare strings.
- ♦ **createParms** The parameters used to configure the new database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [ULCreateParms class](#).

Remarks

To specify a collation, you must first include the appropriate collation data source file from the src\ulcollations\cs (for C# projects) or src\ulcollations\vb.net (for Visual Basic projects) subdirectory of your SQL Anywhere installation directory. Once included in your project, use the collation's Data property to supply the collation data to the CreateDatabase() method.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows CE device then opens a connection to it.

```
' Visual Basic  
Dim openParms As ULConnectionParms = New ULConnectionParms  
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"  
' Assumes file src\ulcollations\vb.net\coll_1250LATIN2.vb is  
' also compiled in the current project  
ULConnection.DatabaseManager.CreateDatabase( _  
    openParms.ToString(), _  
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data, _  
    "" _  
)  
Dim conn As ULConnection = _  
    New ULConnection( openParms.ToString() )  
conn.Open()  
  
// C#  
ULConnectionParms openParms = new ULConnectionParms();  
openParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";  
// Assumes file src\ulcollations\cs\coll_1250LATIN2.cs is  
// also compiled in the current project  
ULConnection.DatabaseManager.CreateDatabase(  
    openParms.ToString(),  
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data,
```

```
""  
);  
ULConnection conn = new ULConnection( openParms.ToString() );  
conn.Open();
```

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“Open method” on page 115](#)

DropDatabase method

Deletes the specified database.

You cannot drop a database that has open connections.

Syntax

Visual Basic

```
Public Sub DropDatabase( _  
    ByVal connString As String _  
)
```

C#

```
public void DropDatabase(  
    string connString  
);
```

Parameters

- ◆ **connString** The parameters for identifying a database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [ULConnectionParms class](#).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows CE device then opens a connection to it.

```
' Visual Basic  
Dim connParms As ULConnectionParms = New ULConnectionParms  
connParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"  
ULConnection.DatabaseManager.DropDatabase( _  
    connParms.ToString() _  
)  
  
// C#  
ULConnectionParms connParms = new ULConnectionParms();  
connParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";  
ULConnection.DatabaseManager.DropDatabase(  
    connParms.ToString()  
);
```

```
ULConnection conn = new ULConnection( openParms.ToString() );  
conn.Open();
```

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“Open method” on page 115](#)

SetActiveSyncListener method

Specifies the listener object used to process ActiveSync calls from the MobiLink provider for ActiveSync.

Syntax**Visual Basic**

```
Public Sub SetActiveSyncListener( _  
    ByVal appClassName As String, _  
    ByVal listener As ULSyncListener _  
)
```

C#

```
public void SetActiveSyncListener(  
    string appClassName,  
    ULSyncListener listener  
);
```

Parameters

- ◆ **appClassName** The unique class name for the application. This is the class name used when the application is registered for use with ActiveSync.
- ◆ **listener** The ULSyncListener object. Use null (Nothing in Visual Basic) to remove the previous listener.

Remarks

The parameter *appClassName* is the unique identifier used to identify the application. The application can only use one *appClassName* at a time. While a listener is registered with a particular *appClassName*, calls to SetServerSyncListener or SetActiveSyncListener with a different *appClassName* fail.

To remove the ActiveSync listener, call SetActiveSyncListener with a null reference (Nothing in Visual Basic) as the *listener* parameter.

To remove all listeners, call SetServerSyncListener with a null reference (Nothing in Visual Basic) for all parameters.

Applications should remove all listeners prior to exiting.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Example

For an example of SetActiveSyncListener, see the [ActiveSyncInvoked method](#).

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“SetServerSyncListener method” on page 183](#)
- ◆ [“SetActiveSyncListener method” on page 182](#)
- ◆ [“ULActiveSyncListener interface” on page 53](#)
- ◆ [“ActiveSyncInvoked method” on page 53](#)

SetServerSyncListener method

Specifies the listener object used to process the specified server synchronization message.

Syntax

Visual Basic

```
Public Sub SetServerSyncListener( _  
    ByVal messageName As String, _  
    ByVal appClassName As String, _  
    ByVal listener As ULServerSyncListener _  
)
```

C#

```
public void SetServerSyncListener(  
    string messageName,  
    string appClassName,  
    ULServerSyncListener listener  
);
```

Parameters

- ◆ **messageName** The name of the message.
- ◆ **appClassName** The unique class name for the application. This is a unique identifier used to identify the application.
- ◆ **listener** The ULServerSyncListener object. Use null (Nothing in Visual Basic) to remove the previous listener.

Remarks

The parameter *appClassName* is the unique identifier used to identify the application. The application may only use one *appClassName* at a time. While a listener is registered with a particular *appClassName*, calls to SetServerSyncListener or SetActiveSyncListener with a different *appClassName* fail.

To remove the listener for a particular message, call SetServerSyncListener with a null reference (Nothing in Visual Basic) as the *listener* parameter.

To remove all listeners, call `SetServerSyncListener` with a null reference (Nothing in Visual Basic) for all parameters.

Applications should remove all listeners before exiting.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Example

For an example of `SetServerSyncListener`, see the [ServerSyncInvoked method](#).

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“SetServerSyncListener method” on page 183](#)
- ◆ [“SetActiveSyncListener method” on page 182](#)
- ◆ [“ServerSyncInvoked method” on page 354](#)

SignalSyncIsComplete method

Signals the MobiLink provider for ActiveSync that an application has completed synchronization.

Syntax

Visual Basic

```
Public Sub SignalSyncIsComplete()
```

C#

```
public void SignalSyncIsComplete();
```

Example

For an example of `SignalSyncIsComplete`, see the [ActiveSyncInvoked method](#).

See also

- ◆ [“ULDatabaseManager class” on page 178](#)
- ◆ [“ULDatabaseManager members” on page 178](#)
- ◆ [“SignalSyncIsComplete method” on page 184](#)

ULDatabaseSchema class

UL Ext.: Represents the schema of an UltraLite.NET database.

Syntax

Visual Basic

NotInheritable Public Class **ULDatabaseSchema**

C#

public sealed class **ULDatabaseSchema**

Remarks

There is no constructor for this class. A ULDatabaseSchema object is attached to a connection as its ULConnection.Schema and is only valid while that connection is open.

See also

- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“Schema property” on page 100](#)

ULDatabaseSchema members

Public properties

Member name	Description
CollationName property	The name of the database's collation sequence.
IsCaseSensitive property	Checks whether the database is case sensitive.
IsOpen property	Whether the database schema is open.
PublicationCount property	The number of publications in the database.
TableCount property	The number of tables in the database.

Public methods

Member name	Description
GetDatabaseProperty method	Returns the value of the specified database property.
GetPublicationName method	Returns the name of the publication identified by the specified publication ID. Publication IDs are not publication masks.
GetPublicationSchema method	Returns the publication schema corresponding to the named publication.

Member name	Description
GetTableCountInPublications method	Returns the number of tables included in the specified publication mask.
GetTableName method	Returns the name of the table identified by the specified table ID.
SetDatabaseOption method	Sets the value for the specified database option.

CollationName property

The name of the database's collation sequence.

Syntax

Visual Basic

Public Readonly Property **CollationName** As String

C#

```
public string CollationName {get;}
```

Property value

A string representing the database's collation sequence.

Remarks

The database collation sequence affects how indexes on tables and result sets are sorted.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“GetDatabaseProperty method” on page 188](#)

IsCaseSensitive property

Checks whether the database is case sensitive.

Syntax

Visual Basic

Public Readonly Property **IsCaseSensitive** As Boolean

C#

```
public bool IsCaseSensitive {get;}
```

Property value

True if the database is case sensitive, and false if the database is case insensitive.

Remarks

Database case sensitivity affects how indexes on tables and result sets are sorted. Case sensitivity also affects how ULConnectionParms.UserID and ULConnectionParms.Password are verified.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“GetDatabaseProperty method” on page 188](#)
- ◆ [“UserID property” on page 131](#)
- ◆ [“Password property” on page 130](#)

IsOpen property

Whether the database schema is open.

Syntax**Visual Basic**

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen {get;}
```

Property value

True if this database schema is currently open, false if this database schema is currently closed.

Remarks

A ULDatabaseSchema object is open only if the connection it is attached to is open.

PublicationCount property

The number of publications in the database.

Syntax**Visual Basic**

Public Readonly Property **PublicationCount** As Integer

C#

public int **PublicationCount** {get;}

Property value

The number of publications in the database or zero if the connection is not open.

Remarks

Publication IDs range from 1 to PublicationCount, inclusively. Publication IDs are not publication masks.

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“GetPublicationName method” on page 191](#)

TableCount property

The number of tables in the database.

Syntax**Visual Basic**

Public Readonly Property **TableCount** As Integer

C#

public int **TableCount** {get;}

Property value

The number of tables in the database or zero if the connection is not open.

Remarks

Table IDs range from 1 to TableCount, inclusively.

Note: Table IDs and counts may change during a schema upgrade. To correctly identify a table, access it by name or refresh the cached IDs and counts after a schema upgrade.

GetDatabaseProperty method

Returns the value of the specified database property.

Syntax

Visual Basic

```
Public Function GetDatabaseProperty( _  
    ByVal name As String _  
) As String
```

C#

```
public string GetDatabaseProperty(  
    string name  
);
```

Parameters

- ♦ **name** The name of the database property whose value you want to obtain. Property names are case insensitive.

Return value

The value of the property as a string.

Remarks

Recognized properties are:

Property	Description
CaseSensitive	The status of the case sensitivity feature. Returns ON if the database is case sensitive. Otherwise, it returns OFF. Database case sensitivity affects how indexes on tables and result sets are sorted. Case sensitivity does not affect how a connection's ULConnectionParms.UserID and ULConnectionParms.Password are verified. User IDs are always case insensitive and passwords are always case sensitive.
CharSet	The character set of the database.
ChecksumLevel	The level of database page checksums enabled for the database.
CollationName	The name of the database's collation sequence.
ConnCount	The number of connections to the database.
date_format	The date format used for string conversions by the database. This format is not necessarily the same as the one used by System.DateTime.
date_order	The date order used for string conversions by the database.
Encryption	The type of encryption applied to the database. Returns None, Simple, AES, or AES_FIPS.

Property	Description
File	The file name of the database.
global_database_id	The value of the global_database_id option used for global autoincrement columns.
MaxHashSize	The default maximum number of bytes to use for index hashing. This property can be set on a per-index basis.
ml_remote_id	The value of the ml_remote_id option used for identifying the database during synchronization.
Name	The name of the database (DBN).
nearest_century	The nearest century used for string conversions by the database.
PageSize	The page size of the database, in bytes.
precision	The floating-point precision used for string conversions by the database.
scale	The minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the database.
time_format	The time format used for string conversions by the database. This format is not necessarily the same as the one used by System.TimeSpan.
timestamp_format	The timestamp format used for string conversions by the database. This format is not necessarily the same as the one used by System.DateTime.
timestamp_increment	The minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“SetDatabaseOption method” on page 193](#)
- ◆ [“CollationName property” on page 186](#)
- ◆ [“UserID property” on page 131](#)
- ◆ [“Password property” on page 130](#)
- ◆ [TimeSpan](#)
- ◆ [DateTime](#)

GetPublicationName method

Returns the name of the publication identified by the specified publication ID. Publication IDs are not publication masks.

Syntax

Visual Basic

```
Public Function GetPublicationName( _  
    ByVal pubID As Integer _  
) As String
```

C#

```
public string GetPublicationName(  
    int pubID  
);
```

Parameters

- ◆ **pubID** The ID of the publication. The value must be in the range [1,PublicationCount].

Return value

The publication name as a string.

Remarks

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“PublicationCount property” on page 187](#)
- ◆ [“PublicationCount property” on page 187](#)

GetPublicationSchema method

Returns the publication schema corresponding to the named publication.

Syntax

Visual Basic

```
Public Function GetPublicationSchema( _  
    ByVal name As String _  
) As ULPublicationSchema
```

C#

```
public ULPublicationSchema GetPublicationSchema(  
    string name  
);
```

Parameters

- ♦ **name** The name of the publication.

Return value

The ULPublicationSchema object representing the named publication.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

See also

- ♦ [“ULDatabaseSchema class” on page 185](#)
- ♦ [“ULDatabaseSchema members” on page 185](#)
- ♦ [“GetPublicationName method” on page 191](#)
- ♦ [“ULPublicationSchema class” on page 312](#)

GetTableCountInPublications method

Returns the number of tables included in the specified publication mask.

Syntax**Visual Basic**

```
Public Function GetTableCountInPublications( _  
    ByVal mask As Integer _  
) As Integer
```

C#

```
public int GetTableCountInPublications(  
    int mask  
);
```

Parameters

- ♦ **mask** The set of publications to check. For information on building publication masks, see [ULPublicationSchema class](#).

Return value

The number of tables included in set of publications.

Remarks

The count will not include tables whose names end in `_nosync`.

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

GetTableName method

Returns the name of the table identified by the specified table ID.

Syntax

Visual Basic

```
Public Function GetTableName( _  
    ByVal tableID As Integer _  
) As String
```

C#

```
public string GetTableName(  
    int tableID  
);
```

Parameters

- ◆ **tableID** The ID of the table. The value must be in range [1,TableCount].

Return value

The table name as a string.

Remarks

Table IDs may change during a schema upgrade. To correctly identify a table, access it by name or refresh the cached IDs after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“TableCount property” on page 188](#)

SetDatabaseOption method

Sets the value for the specified database option.

Syntax

Visual Basic

```
Public Sub SetDatabaseOption( _  
    ByVal name As String, _  
    ByVal value As String _  
)
```

C#

```
public void SetDatabaseOption(  
    string name,  
    string value  
);
```

Parameters

- ◆ **name** The name of the database option. Option names are case insensitive.
- ◆ **value** The new value for the option.

Remarks

Setting a database option results in a commit being performed.

Recognized options are:

Option	Description
global_database_id	The value used for global autoincrement columns. The value must be in the range [0, System.UInt32.MaxValue]. The default is ULConnection.INVALID_DATABASE_ID (used to indicate that the database ID has not been set for the current database).
ml_remote_id	The value used for identifying the database during synchronization. Use a null reference (Nothing in Visual Basic) as the value to remove the ml_remote_id option from the database.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDatabaseSchema class” on page 185](#)
- ◆ [“ULDatabaseSchema members” on page 185](#)
- ◆ [“GetDatabaseProperty method” on page 188](#)
- ◆ [UInt32.MaxValue](#)
- ◆ [“INVALID_DATABASE_ID field” on page 95](#)

ULDataReader class

Represents a read-only bi-directional cursor in an UltraLite database. Cursors are sets of rows from either a table or the result set from a query.

Syntax

Visual Basic

Public Class **ULDataReader**
Inherits MarshalByRefObject

C#

```
public class ULDataReader :  
    MarshalByRefObject
```

Remarks

There is no constructor for ULDataReader. To get a ULDataReader object, execute a ULCommand:

```
' Visual Basic  
Dim cmd As ULCommand = new ULCommand(  
    "SELECT emp_id FROM employee FOR READ ONLY", conn _  
)  
Dim reader As ULDataReader = cmd.ExecuteReader()  
  
// C#  
ULCommand cmd = new ULCommand(  
    "SELECT emp_id FROM employee FOR READ ONLY", conn  
);  
ULDataReader reader = cmd.ExecuteReader();
```

UL Ext.: The ADO.NET standard only requires forward-only motion through the result set, but ULDataReader is bi-directional. ULDataReader's Move methods provide you with full flexibility when moving through results.

ULDataReader is a read-only result set. If you need a more flexible object to manipulate results, use a ULCommand.ExecuteResultSet(), ULCommand.ExecuteTable(), or a ULDataAdapter. ULDataReader retrieves rows as needed, whereas ULDataAdapter must retrieve all rows of a result set before you can carry out any action on the object. For large result sets, this difference gives the ULDataReader a much faster response time.

UL Ext.: All columns of a ULDataReader may be retrieved using GetString.

Implements: System.Data.IDataReader, System.Data.IDataRecord, System.IDisposable

See also

- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ExecuteResultSet method” on page 74](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“GetString method” on page 225](#)

- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

ULDataReader members

Public properties

Member name	Description
Depth property	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property	Returns the number of columns in the cursor.
IsBOF property	UL Ext.: Checks whether the current row position is before the first row.
IsClosed property	Checks whether the cursor is currently open.
IsEOF property	UL Ext.: Checks whether the current row position is after the last row.
Item property	Returns the value of the specified column in its native format. In C#, this property is the indexer for the ULDataReader class.
Item property	Returns the value of the specified named column in its native format. In C#, this property is the indexer for the ULDataReader class.
RecordsAffected property	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property	UL Ext.: Returns the number of rows in the cursor.
Schema property	UL Ext.: Holds the schema of this cursor.

Public methods

Member name	Description
Close method	Closes the cursor.
Dispose method	Releases the unmanaged resources used by the ULDataReader and optionally releases the managed resources.
GetBoolean method	Returns the value for the specified column as a System.Boolean.
GetByte method	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
GetBytes method	Copies a subset of the value for the specified ULDbType.LongBinary column, beginning at the specified offset, to the specified offset of the destination System.Byte array.

Member name	Description
GetBytes method	UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.
GetChar method	This method is not supported in UltraLite.NET.
GetChars method	Copies a subset of the value for the specified ULDbType.LongVarChar column, beginning at the specified offset, to the specified offset of the destination System.Char array.
GetData method	This method is not supported in UltraLite.NET.
GetDataTypeName method	Returns the name of the specified column's provider data type.
GetDateTime method	Returns the value for the specified column as a System.DateTime with millisecond accuracy.
GetDecimal method	Returns the value for the specified column as a System.Decimal.
GetDouble method	Returns the value for the specified column as a System.Double.
GetFieldType method	Returns the System.Type most appropriate for the specified column.
GetFloat method	Returns the value for the specified column as a System.Single.
GetGuid method	Returns the value for the specified column as a UUID (System.Guid).
GetInt16 method	Returns the value for the specified column as a System.Int16.
GetInt32 method	Returns the value for the specified column as an Int32.
GetInt64 method	Returns the value for the specified column as an Int64.
GetName method	Returns the name of the specified column.
GetOrdinal method	Returns the column ID of the named column.
GetSchemaTable method	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader.
GetString method	Returns the value for the specified column as a System.String.
GetTimeSpan method	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method	Returns the value for the specified column as a System.UInt16.
GetUInt32 method	Returns the value for the specified column as a UInt32.
GetUInt64 method	Returns the value for the specified column as a System.UInt64.
GetValue method	Returns the value of the specified column in its native format.

Member name	Description
GetValues method	Returns all the column values for the current row.
IsDBNull method	Checks whether the value from the specified column is NULL.
MoveAfterLast method	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method	UL Ext.: Positions the cursor to the first row of the cursor.
MoveLast method	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method	UL Ext.: Positions the cursor relative to the current row.
NextResult method	Advances the ULDataReader to the next result when reading the results of batch SQL statements.
Read method	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.

Protected methods

Member name	Description
Finalize method	Releases the unmanaged resources used by the ULDataReader.

Depth property

Returns the depth of nesting for the current row. The outermost table has a depth of zero.

Syntax

Visual Basic

NotOverridable Public Readonly Property **Depth** As Integer _
Implements IDataReader.Depth

C#

```
public int Depth {get;}
```

Property value

All UltraLite.NET result sets have a depth of zero.

Exceptions

- ◆ [ULException class](#) - The ULDataReader is not opened.

Implements

[IDataReader.Depth](#)

FieldCount property

Returns the number of columns in the cursor.

Syntax

Visual Basic

NotOverridable Public Readonly Property **FieldCount** As Integer _
Implements IDataRecord.FieldCount

C#

```
public int FieldCount {get;}
```

Return value

The number of columns in the cursor as an integer. Returns 0 if the cursor is closed.

Remarks

This method is identical to the `ULCursorSchema.ColumnCount` method.

Implements

[IDataRecord.FieldCount](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“ColumnCount property” on page 145](#)

IsBOF property

UL Ext.: Checks whether the current row position is before the first row.

Syntax

Visual Basic

Public Readonly Property **IsBOF** As Boolean

C#

```
public bool IsBOF {get;}
```

Property value

True if the current row position is before the first row, false otherwise.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

IsClosed property

Checks whether the cursor is currently open.

Syntax

Visual Basic

```
NotOverridable Public Readonly Property IsClosed As Boolean _  
    Implements IDataReader.IsClosed
```

C#

```
public bool IsClosed {get;}
```

Property value

True if the cursor is currently open, false if the cursor is closed.

Implements

[IDataReader.IsClosed](#)

IsEOF property

UL Ext.: Checks whether the current row position is after the last row.

Syntax

Visual Basic

```
Public Readonly Property IsEOF As Boolean
```

C#

```
public bool IsEOF {get;}
```

Property value

True if the current row position is after the last row, false otherwise.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

Item property

Returns the value of the specified column in its native format. In C#, this property is the indexer for the ULDataReader class.

Syntax

Visual Basic

```
NotOverridable Public Readonly Property Item As Object _  
    Implements IDataRecord.Item
```

C#

```
public object Item {get;}
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

This method is identical in functionality to the ULDataReader.GetValue(int) method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.Item](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“Item property” on page 201](#)
- ◆ [“GetValue method” on page 229](#)
- ◆ [“FieldCount property” on page 199](#)

Item property

Returns the value of the specified named column in its native format. In C#, this property is the indexer for the ULDataReader class.

Syntax

Visual Basic

NotOverridable Public Readonly Property **Item** As Object _
Implements IDataRecord.Item

C#

public object **Item** {get;}

Parameters

- ◆ **name** The name of the column.

Return value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

When accessing columns multiple times, it is more efficient to access columns by column ID than by name.

This method is equivalent to:

```
dataReader.GetValue( dataReader.GetOrdinal( name ) )
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.Item](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“Item property” on page 201](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetValue method” on page 229](#)
- ◆ [“GetFieldType method” on page 216](#)

RecordsAffected property

Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.

Syntax

Visual Basic

NotOverridable Public Readonly Property **RecordsAffected** As Integer _
Implements IDataReader.RecordsAffected

C#

```
public int RecordsAffected {get;}
```

Property value

The number of rows changed, inserted, or deleted by execution of the SQL statement.

Implements

[IDataReader.RecordsAffected](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [CommandType.TableDirect](#)

RowCount property

UL Ext.: Returns the number of rows in the cursor.

Syntax**Visual Basic**

Public Readonly Property **RowCount** As Integer

C#

```
public int RowCount {get;}
```

Property value

The number of rows in the cursor.

Remarks

One use for RowCount is to decide when to delete old rows to save space. Old rows can be deleted from the UltraLite database without being deleted from the consolidated database using the `ULConnection.StopSynchronizationDelete` method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“StartSynchronizationDelete method” on page 117](#)
- ◆ [“StopSynchronizationDelete method” on page 117](#)

Schema property

UL Ext.: Holds the schema of this cursor.

Syntax

Visual Basic

Public Readonly Property **Schema** As ULCursorSchema

C#

public ULCursorSchema **Schema** {get;}

Property value

For result sets, the ULResultSetSchema object representing the schema of the result set. For tables, the ULTableSchema object representing the schema of the table.

Remarks

This property represents the complete schema of the cursor, including UltraLite.NET extended information which is not represented in the results from ULDataReader.GetSchemaTable.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“GetSchemaTable method” on page 223](#)
- ◆ [“ULResultSetSchema class” on page 342](#)

Close method

Closes the cursor.

Syntax

Visual Basic

NotOverridable Public Sub **Close()** _
Implements IDataReader.Close

C#

public void **Close()**;

Remarks

It is not an error to close a cursor that is already closed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements[IDataReader.Close](#)**Dispose method**

Releases the unmanaged resources used by the ULDataReader and optionally releases the managed resources.

Syntax**Visual Basic**

NotOverridable Public Sub **Dispose()** _
Implements IDisposable.Dispose

C#

public void **Dispose()**;

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements[IDisposable.Dispose](#)**Finalize method**

Releases the unmanaged resources used by the ULDataReader.

Syntax**Visual Basic**

Overrides Protected Sub **Finalize()**

C#

protected override void **Finalize()**;

GetBoolean method

Returns the value for the specified column as a System.Boolean.

Syntax**Visual Basic**

NotOverridable Public Function **GetBoolean**(_
ByVal *columnID* As Integer _

```
) As Boolean _  
    Implements IDataRecord.GetBoolean
```

C#

```
public bool GetBoolean(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Boolean.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetBoolean](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Boolean](#)
- ◆ [“FieldCount property” on page 199](#)

GetByte method

Returns the value for the specified column as an unsigned 8-bit value (System.Byte).

Syntax**Visual Basic**

```
NotOverridable Public Function GetByte( _  
    ByVal columnID As Integer _  
) As Byte _  
    Implements IDataRecord.GetByte
```

C#

```
public byte GetByte(  
    int columnID  
);
```


Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Byte.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetByte](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Byte](#)
- ◆ [“FieldCount property” on page 199](#)

GetBytes method

Copies a subset of the value for the specified ULDbType.LongBinary column, beginning at the specified offset, to the specified offset of the destination System.Byte array.

Syntax**Visual Basic**

```
Overloads NotOverridable Public Function GetBytes( _  
    ByVal columnID As Integer, _  
    ByVal srcOffset As Long, _  
    ByVal dst As Byte(), _  
    ByVal dstOffset As Integer, _  
    ByVal count As Integer _  
) As Long _  
    Implements IDataRecord.GetBytes
```

C#

```
public long GetBytes(  
    int columnID,  
    long srcOffset,  
    byte[] dst,  
    int dstOffset,  
    int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **srcOffset** The start position in the column value. Zero is the beginning of the value.
- ◆ **dst** The destination array.
- ◆ **dstOffset** The start position in the destination array.
- ◆ **count** The number of bytes to be copied.

Return value

The actual number of bytes copied.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), GetBytes returns the length of the field in bytes.

The bytes at position *srcOffset* through *srcOffset+count-1* of the value are copied into positions *dstOffset* through *dstOffset+count-1*, respectively, of the destination array. If the end of the value is encountered before *count* bytes are copied, the remainder of the destination array is left unchanged.

If any of the following is true, a *ULException* with code *ULSQLCode.SQLE_INVALID_PARAMETER* is thrown and the destination is not modified:

- ◆ *srcOffset* is negative.
- ◆ *dstOffset* is negative.
- ◆ *count* is negative.
- ◆ *dstOffset+count* is greater than *dst.Length*.

For other errors, a *ULException* with the appropriate error code is thrown.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetBytes](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetBytes method” on page 209](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [Byte](#)

- ◆ [“ULException class” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“FieldCount property” on page 199](#)

GetBytes method

UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.

Syntax

Visual Basic

```
Overloads Public Function GetBytes( _  
    ByVal columnID As Integer _  
) As Byte()
```

C#

```
public byte[] GetBytes(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an array of System.Bytes.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetBytes method” on page 207](#)
- ◆ [Byte](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)

GetChar method

This method is not supported in UltraLite.NET.

Syntax

Visual Basic

```
NotOverridable Public Function GetChar( _  
    ByVal columnID As Integer _  
) As Char _  
    Implements IDataRecord.GetChar
```

C#

```
public char GetChar(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

This method is not supported in UltraLite.NET.

Exceptions

- ◆ [ULException class](#) - This method is not supported in UltraLite.NET.

Implements

[IDataRecord.GetChar](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetString method” on page 225](#)
- ◆ [“FieldCount property” on page 199](#)

GetChars method

Copies a subset of the value for the specified ULDbType.LongVarchar column, beginning at the specified offset, to the specified offset of the destination System.Char array.

Syntax

Visual Basic

```
NotOverridable Public Function GetChars( _  
    ByVal columnID As Integer, _  
    ByVal srcOffset As Long, _  
    ByVal dst As Char(), _  
    ByVal dstOffset As Integer, _  
    ByVal count As Integer _
```

```
) As Long _  
    Implements IDataRecord.GetChars
```

C#

```
public long GetChars(  
    int columnID,  
    long srcOffset,  
    char[] dst,  
    int dstOffset,  
    int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **srcOffset** The start position in the column value. Zero is the beginning of the value.
- ◆ **dst** The destination array.
- ◆ **dstOffset** The start position in the destination array.
- ◆ **count** The number of characters to be copied.

Return value

The actual number of characters copied.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), GetChars returns the length of the field in characters.

The characters at position *srcOffset* through *srcOffset+count-1* of the value are copied into positions *dstOffset* through *dstOffset+count-1*, respectively, of the destination array. If the end of the value is encountered before *count* characters are copied, the remainder of the destination array is left unchanged.

If any of the following is true, a *ULException* with code *ULSQLCode.SQLE_INVALID_PARAMETER* is thrown and the destination is not modified:

- ◆ *srcOffset* is negative.
- ◆ *dstOffset* is negative.
- ◆ *count* is negative.
- ◆ *dstOffset+count* is greater than *dst.Length*.

For other errors, a *ULException* with the appropriate error code is thrown.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetChars](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [System.Char](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“FieldCount property” on page 199](#)

GetData method

This method is not supported in UltraLite.NET.

Syntax

Visual Basic

```
NotOverridable Public Function GetData( _  
    ByVal i As Integer _  
) As IDataReader _  
    Implements IDataRecord.GetData
```

C#

```
public IDataReader GetData(  
    int i  
);
```

Parameters

- ◆ **i** An integer value.

Return value

This method is not supported in UltraLite.NET.

Exceptions

- ◆ [ULException class](#) - This method is not supported in UltraLite.NET.

Implements

[IDataRecord.GetData](#)

GetDataTypeName method

Returns the name of the specified column's provider data type.

Syntax

Visual Basic

```
NotOverridable Public Function GetDataTypeName( _  
    ByVal columnID As Integer _  
) As String _  
    Implements IDataRecord.GetDataTypeName
```

C#

```
public string GetDataTypeName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

A string corresponding to the column's ULDbType.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetDataTypeName](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“ULDbType enumeration” on page 238](#)

GetDateTime method

Returns the value for the specified column as a System.DateTime with millisecond accuracy.

Syntax

Visual Basic

```
NotOverridable Public Function GetDateTime( _  
    ByVal columnID As Integer _
```

```
) As Date _  
    Implements IDataRecord.GetDate
```

C#

```
public DateTime GetDateTime(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.DateTime.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetDateTime](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [DateTime](#)
- ◆ [“FieldCount property” on page 199](#)

GetDecimal method

Returns the value for the specified column as a System.Decimal.

Syntax

Visual Basic

```
NotOverridable Public Function GetDecimal( _  
    ByVal columnID As Integer _  
) As Decimal _  
    Implements IDataRecord.GetDecimal
```

C#

```
public decimal GetDecimal(  
    int columnID  
);
```


Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Decimal.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetDecimal](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Decimal](#)
- ◆ [“FieldCount property” on page 199](#)

GetDouble method

Returns the value for the specified column as a System.Double.

Syntax**Visual Basic**

```
NotOverridable Public Function GetDouble( _  
    ByVal columnID As Integer _  
) As Double _  
    Implements IDataRecord.GetDouble
```

C#

```
public double GetDouble(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Double.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetDouble](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Double](#)
- ◆ [“FieldCount property” on page 199](#)

GetFieldType method

Returns the System.Type most appropriate for the specified column.

Syntax

Visual Basic

```
NotOverridable Public Function GetFieldType( _  
    ByVal columnID As Integer _  
) As Type _  
    Implements IDataRecord.GetFieldType
```

C#

```
public Type GetFieldType(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

A System.Type value for the column.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetFieldType](#)

See also

- ◆ [“ULDataReader class” on page 195](#)

- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetDataTypeName method” on page 213](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [Type](#)
- ◆ [“FieldCount property” on page 199](#)

GetFloat method

Returns the value for the specified column as a System.Single.

Syntax

Visual Basic

```
NotOverridable Public Function GetFloat( _  
    ByVal columnID As Integer _  
) As Single _  
    Implements IDataRecord.GetFloat
```

C#

```
public float GetFloat(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Single.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetFloat](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Single](#)
- ◆ [“FieldCount property” on page 199](#)

GetGuid method

Returns the value for the specified column as a UUID (System.Guid).

Syntax

Visual Basic

```
NotOverridable Public Function GetGuid( _  
    ByVal columnID As Integer _  
) As Guid _  
    Implements IDataRecord.GetGuid
```

C#

```
public Guid GetGuid(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a Guid.

Remarks

This method is only valid for columns of type `ULDbType.UniqueIdentifier` or for columns of type `ULDbType.Binary` with length 16.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetGuid](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [“GetColumnSize method” on page 150](#)
- ◆ [Guid](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“FieldCount property” on page 199](#)

GetInt16 method

Returns the value for the specified column as a System.Int16.

Syntax

Visual Basic

```
NotOverridable Public Function GetInt16( _  
    ByVal columnID As Integer _  
) As Short _  
    Implements IDataRecord.GetShort
```

C#

```
public short GetInt16(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an System.Int16.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetInt16](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Int16](#)
- ◆ [“FieldCount property” on page 199](#)

GetInt32 method

Returns the value for the specified column as an Int32.

Syntax

Visual Basic

```
NotOverridable Public Function GetInt32( _  
    ByVal columnID As Integer _
```

```
) As Integer _  
    Implements IDataRecord.GetInteger
```

C#

```
public int GetInt32(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an Int32.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetInt32](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Int32](#)
- ◆ [“FieldCount property” on page 199](#)

GetInt64 method

Returns the value for the specified column as an Int64.

Syntax

Visual Basic

```
NotOverridable Public Function GetInt64( _  
    ByVal columnID As Integer _  
) As Long _  
    Implements IDataRecord.GetLong
```

C#

```
public long GetInt64(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an Int64.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetInt64](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Int64](#)
- ◆ [“FieldCount property” on page 199](#)

GetName method

Returns the name of the specified column.

Syntax

Visual Basic

```
NotOverridable Public Function GetName( _  
    ByVal columnID As Integer _  
) As String _  
    Implements IDataRecord.GetName
```

C#

```
public string GetName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

This method is identical to the `ULCursorSchema.GetColumnName` method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetName](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“GetSchemaTable method” on page 223](#)
- ◆ [“GetColumnName method” on page 147](#)
- ◆ [“FieldCount property” on page 199](#)

GetOrdinal method

Returns the column ID of the named column.

Syntax

Visual Basic

```
NotOverridable Public Function GetOrdinal( _  
    ByVal columnName As String _  
) As Integer _  
    Implements IDataRecord.GetOrdinal
```

C#

```
public int GetOrdinal(  
    string columnName  
);
```

Parameters

- ◆ **columnName** The name of the column.

Return value

The column ID of the named column.

Remarks

Column IDs range from 0 to `ULDataReader.FieldCount-1`, inclusively.

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and counts may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

This method is identical to the `ULCursorSchema.GetColumnID` method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetOrdinal](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetSchemaTable method” on page 223](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“GetColumnID method” on page 146](#)

GetSchemaTable method

Returns a `System.Data.DataTable` that describes the column metadata of the `ULDataReader`.

Syntax

Visual Basic

NotOverridable Public Function **GetSchemaTable()** As DataTable _
Implements IDataReader.GetSchemaTable

C#

public DataTable **GetSchemaTable();**

Return value

A `System.Data.DataTable` describing the schema of each column in the `ULDataReader`.

Remarks

The `GetSchemaTable` method returns metadata about each column in the following order:

DataTable column	Description
ColumnName	The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned. Note that in result sets, not all columns have names and not all column names are unique.

DataTable column	Description
ColumnOrdinal	The ID of the column. The value is in the range [0,FieldCount-1].
ColumnSize	For sized columns, the maximum length of a value in the column. For other columns, this is the size in bytes of the data type.
NumericPrecision	The precision of a numeric column (ProviderType ULDbType.Decimal or ULDbType.Numeric) or DBNull if the column is not numeric.
NumericScale	The scale of a numeric column (ProviderType ULDbType.Decimal or ULDbType.Numeric) or DBNull if the column is not numeric.
IsUnique	True if the column is a non-computed unique column in the table (BaseTableName) it is taken from.
IsKey	True if the column is one of a set of columns in the result set that taken together from a unique key for the result set. The set of columns with IsKey set to true does not need to be the minimal set that uniquely identifies a row in the result set.
BaseCatalogName	The name of the catalog in the database that contains the column. For UltraLite.NET, this value is always DBNull.
BaseColumnName	The original name of the column in the table BaseTableName of the database or DBNull if the column is computed or if this information cannot be determined.
BaseSchemaName	The name of the schema in the database that contains the column. For UltraLite.NET, this value is always DBNull.
BaseTableName	The name of the table in the database that contains the column, or DBNull if column is computed or if this information cannot be determined.
DataType	The .NET data type that is most appropriate for this type of column.
AllowDBNull	True if the column is nullable, false if the column is not nullable or if this information cannot be determined.
ProviderType	The ULDbType of the column.
IsIdentity	True if the column is an identity column, false if it is not an identity column. For UltraLite.NET, this value is always false.
IsAutoIncrement	True if the column is an autoincrement or global autoincrement column, false otherwise (or if this information cannot be determined).
IsRowVersion	True if the column contains a persistent row identifier that cannot be written to, and has no meaningful value except to identity the row. For UltraLite.NET, this value is always false.
IsLong	True if the column is a ULDbType.LongVarchar or a ULDbType.LongBinary column, false otherwise.

DataTable column	Description
IsReadOnly	True if the column is read-only, false if the column is modifiable or if its access cannot be determined.
IsAliased	True if the column name is an alias, false if it is not an alias.
IsExpression	True if the column is an expression, false if it is a column value.

Implements

[IDataReader.GetSchemaTable](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [DataTable](#)

GetString method

Returns the value for the specified column as a System.String.

Syntax

Visual Basic

```
NotOverridable Public Function GetString( _
    ByVal columnID As Integer _
) As String _
    Implements IDataRecord.GetString
```

C#

```
public string GetString(
    int columnID
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.String.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetString](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [String](#)
- ◆ [“FieldCount property” on page 199](#)

GetTimeSpan method

Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.

Syntax

Visual Basic

```
Public Function GetTimeSpan( _  
    ByVal columnID As Integer _  
) As TimeSpan
```

C#

```
public TimeSpan GetTimeSpan(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.TimeSpan.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [TimeSpan](#)

- ◆ [“FieldCount property” on page 199](#)

GetUInt16 method

Returns the value for the specified column as a System.UInt16.

Syntax

Visual Basic

```
Public Function GetUInt16( _  
    ByVal columnID As Integer _  
) As UInt16
```

C#

```
public ushort GetUInt16(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an System.UInt16.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt16](#)
- ◆ [“FieldCount property” on page 199](#)

GetUInt32 method

Returns the value for the specified column as a UInt32.

Syntax

Visual Basic

```
Public Function GetUInt32( _  
    ByVal columnID As Integer _  
) As UInt32
```

C#

```
public uint GetUInt32(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an UInt32.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt32](#)
- ◆ [“FieldCount property” on page 199](#)

GetUInt64 method

Returns the value for the specified column as a System.UInt64.

Syntax

Visual Basic

```
Public Function GetUInt64(  
    ByVal columnID As Integer  
) As UInt64
```

C#

```
public ulong GetUInt64(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.UInt64

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt64](#)
- ◆ [“FieldCount property” on page 199](#)

GetValue method

Returns the value of the specified column in its native format.

Syntax

Visual Basic

```
NotOverridable Public Function GetValue( _  
    ByVal columnID As Integer _  
) As Object _  
    Implements IDataRecord.GetValue
```

C#

```
public object GetValue(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

This method is identical in functionality to the `ULDataReader.this[int]`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.GetValue](#)

See also

- ◆ [“ULDataReader class” on page 195](#)

- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“Item property” on page 201](#)

GetValues method

Returns all the column values for the current row.

Syntax

Visual Basic

```
NotOverridable Public Function GetValues( _  
    ByVal values As Object() _  
) As Integer _  
    Implements IDataRecord.GetValues
```

C#

```
public int GetValues(  
    object[] values  
);
```

Parameters

- ◆ **values** The array of System.Objects to hold the entire row.

Return value

The number of column values retrieved. If the length of the array is greater than the number of columns (ULDataReader.FieldCount), only FieldCount items are retrieved and the rest of the array is left unchanged.

Remarks

For most applications, the GetValues method provides an efficient means for retrieving all columns, rather than retrieving each column individually.

You can pass an System.Object array that contains fewer than the number of columns contained in the resulting row. Only the amount of data the System.Object array holds is copied to the array. You can also pass an System.Object array whose length is more than the number of columns contained in the resulting row.

This method returns DBNull for NULL database columns. For other columns, it returns the value of the column in its native format.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.
- ◆ [System.ArgumentNullException](#) - The *values* array is NULL or has zero length.

Implements

[IDataRecord.GetValues](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetValue method” on page 229](#)
- ◆ [Object](#)

IsDBNull method

Checks whether the value from the specified column is NULL.

Syntax

Visual Basic

```
NotOverridable Public Function IsDBNull( _  
    ByVal columnID As Integer _  
) As Boolean _  
    Implements IDataRecord.IsDBNull
```

C#

```
public bool IsDBNull(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

True if value is NULL, false if value is not NULL.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataRecord.IsDBNull](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)

- ◆ [“FieldCount property” on page 199](#)

MoveAfterLast method

UL Ext.: Positions the cursor to after the last row of the cursor.

Syntax

Visual Basic

```
Public Sub MoveAfterLast()
```

C#

```
public void MoveAfterLast();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

MoveBeforeFirst method

UL Ext.: Positions the cursor to before the first row of the cursor.

Syntax

Visual Basic

```
Public Sub MoveBeforeFirst()
```

C#

```
public void MoveBeforeFirst();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

MoveFirst method

UL Ext.: Positions the cursor to the first row of the cursor.

Syntax

Visual Basic

```
Public Function MoveFirst() As Boolean
```

C#

```
public bool MoveFirst();
```

Return value

True if successful, false otherwise. For example, the method fails if there are no rows.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

MoveLast method

UL Ext.: Positions the cursor to the last row of the cursor.

Syntax**Visual Basic**

Public Function **MoveLast()** As Boolean

C#

public bool **MoveLast();**

Return value

True if successful, false otherwise. For example, the method fails if there are no rows.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

MoveNext method

UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.

Syntax**Visual Basic**

Public Function **MoveNext()** As Boolean

C#

public bool **MoveNext();**

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

Remarks

This method is identical to the ULDataReader.Read method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“Read method” on page 235](#)

MovePrevious method

UL Ext.: Positions the cursor to the previous row or before the first row.

Syntax**Visual Basic**

Public Function **MovePrevious()** As Boolean

C#

public bool **MovePrevious();**

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

MoveRelative method

UL Ext.: Positions the cursor relative to the current row.

Syntax**Visual Basic**

Public Function **MoveRelative**(_
 ByVal *offset* As Integer _
) As Boolean

C#

public bool **MoveRelative**(
 int *offset*
);

Parameters

- ◆ **offset** The number of rows to move. Negative values correspond to moving backward.

Return value

True if successful, false otherwise. For example, the method fails if it positions beyond the first or last row.

Remarks

If the row does not exist, the method returns false, and the cursor position is after the last row (ULDataReader.IsEOF) if *offset* is positive, and before the first row (ULDataReader.IsBOF) if the *offset* is negative.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“IsEOF property” on page 200](#)
- ◆ [“IsBOF property” on page 199](#)

NextResult method

Advances the ULDataReader to the next result when reading the results of batch SQL statements.

Syntax**Visual Basic**

NotOverridable Public Function **NextResult()** As Boolean _
Implements IDataReader.NextResult

C#

```
public bool NextResult();
```

Return value

True if there are more result sets, false otherwise. For UltraLite.NET, always returns false.

Remarks

UL Ext.: UltraLite.NET does not support batches of SQL statements, hence the ULDataReader is always positioned on the first and only result set. Calling NextResult has no effect.

Exceptions

- ◆ [ULException class](#) - The ULDataReader is not opened.

Implements

[IDataReader.NextResult](#)

Read method

Positions the cursor to the next row, or after the last row if the cursor was already on the last row.

Syntax

Visual Basic

NotOverridable Public Function **Read()** As Boolean _
Implements IDataReader.Read

C#

public bool **Read();**

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

Remarks

This method is identical to the `ULDataReader.MoveNext` method.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Implements

[IDataReader.Read](#)

See also

- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULDataReader members” on page 196](#)
- ◆ [“MoveNext method” on page 233](#)

ULDateOrder enumeration

UL Ext.: Enumerates the date orders that a database can support.

Syntax

Visual Basic

Public Enum **ULDateOrder**

C#

public enum **ULDateOrder**

Member name

Member name	Description
DMY	Day of the month followed by month, followed by year.
MDY	Month followed by day of the month, followed by year.
YMD	Year followed by month, followed by day of the month.

See also

- ♦ [“DateOrder property” on page 137](#)

ULDbType enumeration

Enumerates the UltraLite.NET database data types.

Syntax

Visual Basic

Public Enum **ULDbType**

C#

public enum **ULDbType**

Remarks

The table below lists which .NET types are compatible with each ULDbType. In the case of integral types, table columns can always be set using smaller integer types, but can also be set using larger types as long as the actual value is within the range of the type.

ULDbType	Compatible .NET type	C# built-in type	Visual Basic built-in type
Binary, VarBinary	System.Byte[], or System.Guid if size is 16	byte[]	Byte()
Bit	System.Boolean	bool	Boolean
Char, VarChar	System.String	String	String
Date	System.DateTime	DateTime No built-in type.	Date
Double	System.Double	double	Double
LongBinary	System.Byte[]	byte[]	Byte()
LongVarchar	System.String	String	String
Decimal, Numeric	System.String	decimal	Decimal
Float, Real	System.Single	float	Single
BigInt	System.Int64	long	Long
Integer	System.Int32	int	Integer
SmallInt	System.Int16	short	Short
Time	System.TimeSpan	TimeSpan No built-in type.	TimeSpan No built-in type.

ULDbType	Compatible .NET type	C# built-in type	Visual Basic built-in type
DateTime , TimeStamp	System.DateTime	DateTime No built-in type.	Date
TinyInt	System.Byte	byte	Byte
UnsignedBigInt	System.UInt64	ulong	UInt64 No built-in type.
UnsignedInt	System.UInt32	uint	UInt32 No built-in type.
UnsignedSmall-Int	System.UInt16	ushort	UInt16 No built-in type.
UniqueIdentifier	System.Guid	Guid No built-in type.	Guid No built-in type.

Binary columns of length 16 are fully compatible with the UniqueIdentifier type.

Member name

Member name	Description
BigInt	Signed 64-bit integer.
Binary	Binary data, with a specified maximum length. The enumeration values Binary and VarBinary are aliases of each other.
Bit	1-bit flag.
Char	Character data, with a specified length. In UltraLite.NET, this type always supports Unicode characters. The types Char and VarChar are fully compatible.
Date	Date information.
DateTime	Timestamp information (date, time). The enumeration values Date-Time and TimeStamp are aliases of each other.
Decimal	Exact numerical data, with a specified precision and scale. The enumeration values Decimal and Numeric are aliases of each other.
Double	Double precision floating-point number (8 bytes).
Float	Single precision floating-point number (4 bytes). The enumeration values Float and Real are aliases of each other.
Integer	Unsigned 32-bit integer.

Member name	Description
LongBinary	Binary data, with variable length.
LongVarchar	Character data, with variable length. In UltraLite.NET, this type always supports Unicode characters.
Numeric	Exact numerical data, with a specified precision and scale. The enumeration values Decimal and Numeric are aliases of each other.
Real	Single precision floating-point number (4 bytes). The enumeration values Float and Real are aliases of each other.
SmallInt	Signed 16-bit integer.
Time	Time information.
TimeStamp	Timestamp information (date, time). The enumeration values Date-Time and TimeStamp are aliases of each other.
TinyInt	Unsigned 8-bit integer.
UniqueIdentifier	Universally Unique Identifier (UUID/GUID).
UnsignedBigInt	Unsigned 64-bit integer.
UnsignedInt	Unsigned 32-bit integer.
UnsignedSmallInt	Unsigned 16-bit integer.
VarBinary	Binary data, with a specified maximum length. The enumeration values Binary and VarBinary are aliases of each other.
VarChar	Character data, with a specified maximum length. In UltraLite.NET, this type always supports Unicode characters. The types Char and VarChar are fully compatible.

See also

- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetDataTypeName method” on page 213](#)
- ◆ [“GetColumnULDbType method” on page 151](#)
- ◆ [Byte](#)
- ◆ [Guid](#)
- ◆ [Boolean](#)
- ◆ [String](#)
- ◆ [DateTime](#)
- ◆ [Single](#)
- ◆ [Int64](#)
- ◆ [Int32](#)
- ◆ [Int16](#)
- ◆ [TimeSpan](#)
- ◆ [UInt64](#)

- ◆ UInt32
- ◆ UInt16

ULException class

Represents a SQL error returned by the UltraLite.NET database.

Syntax

Visual Basic

NotInheritable Public Class **ULException**
Inherits ApplicationException

C#

```
public sealed class ULException :  
    ApplicationException
```

Remarks

The SQL code denoting the error is returned in the NativeError.

This class is not serializable under the .NET Compact Framework.

See also

- ♦ [“ULException members” on page 242](#)
- ♦ [“NativeError property” on page 244](#)

ULException members

Public constructors

Member name	Description
ULException constructor	Creates a ULException with the given error code.

Public properties

Member name	Description
HelpLink (inherited from Exception)	Gets or sets a link to the help file associated with this exception.
InnerException (inherited from Exception)	Gets the System.Exception instance that caused the current exception.
Message (inherited from Exception)	Gets a message that describes the current exception.
NativeError property	Returns the SQL code returned by the database.
Source property	Returns the name of the provider that generated the error.

Member name	Description
StackTrace (inherited from Exception)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
TargetSite (inherited from Exception)	Gets the method that throws the current exception.

Public methods

Member name	Description
GetBaseException (inherited from Exception)	When overridden in a derived class, returns the System.Exception that is the root cause of one or more subsequent exceptions.
GetObjectData method	Populates a SerializationInfo with the data needed to serialize this UException.
ToString (inherited from Exception)	Creates and returns a string representation of the current exception.

UException constructor

Creates a UException with the given error code.

Syntax

Visual Basic

```
Overloads Public Sub New( _
    ByVal code As ULSQLCode, _
    ByVal s1 As String, _
    ByVal s2 As String, _
    ByVal s3 As String _
)
```

C#

```
public UException(
    ULSQLCode code,
    string s1,
    string s2,
    string s3
);
```

Parameters

- ◆ **code** The code of the exception.
- ◆ **s1** The first string for the formatted message.
- ◆ **s2** The second string for the formatted message.
- ◆ **s3** The third string for the formatted message.

Remarks

The message string corresponding to the specified `iAnywhere.Data.UltraLite.ULSQLCode` is retrieved from the **iAnywhere.Data.UltraLite.resources** assembly. Resources are searched for, by culture, using the following order: `CultureInfo.CurrentUICulture`, then `CultureInfo.CurrentCulture`, and finally culture "EN".

See also

- ◆ [“ULException class” on page 242](#)
- ◆ [“ULException members” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [System.Globalization.CultureInfo.CurrentUICulture](#)
- ◆ [System.Globalization.CultureInfo.CurrentCulture](#)

NativeError property

Returns the SQL code returned by the database.

Syntax

Visual Basic

Public Readonly Property **NativeError** As ULSQLCode

C#

```
public ULSQLCode NativeError {get;}
```

Property value

The ULSQLCode value returned by the database.

See also

- ◆ [“ULException class” on page 242](#)
- ◆ [“ULException members” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)

Source property

Returns the name of the provider that generated the error.

Syntax

Visual Basic

Public Readonly Property **Source** As String

C#

```
public string Source {get;}
```

Property value

The string value identifying UltraLite.NET as the provider.

GetObjectData method

Populates a *SerializationInfo* with the data needed to serialize this *ULException*.

Syntax**Visual Basic**

```
Overrides Public Sub GetObjectData( _  
    ByVal info As System.Runtime.Serialization.SerializationInfo, _  
    ByVal context As System.Runtime.Serialization.StreamingContext _  
) _  
    Implements ISerializable.GetObjectData
```

C#

```
public override void GetObjectData(  
    System.Runtime.Serialization.SerializationInfo info,  
    System.Runtime.Serialization.StreamingContext context  
);
```

Parameters

- ◆ **info** The *SerializationInfo* to populate with data.
- ◆ **context** The destination for this serialization.

Remarks

This method is not supported under the .NET Compact Framework.

Implements

[ISerializable.GetObjectData](#)

ULFileTransfer class

UL Ext.: Transfers a file from a remote database using the MobiLink server.

Syntax

Visual Basic

NotInheritable Public Class **ULFileTransfer**

C#

public sealed class **ULFileTransfer**

Remarks

You do not need a database connection to perform a file transfer, however, if your application will be using an UltraLite database with the UltraLite Engine runtime, you must set `ULDatabaseManager.RuntimeType` to the appropriate value before using this API or any other UltraLite.NET API.

To transfer a file you must set the `ULFileTransfer.FileName`, `ULFileTransfer.Stream`, `ULFileTransfer.UserName`, and `ULFileTransfer.Version`.

See also

- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“RuntimeType property” on page 179](#)
- ◆ [“FileName property” on page 251](#)
- ◆ [“Stream property” on page 254](#)
- ◆ [“UserName property” on page 256](#)
- ◆ [“Version property” on page 257](#)

ULFileTransfer members

Public constructors

Member name	Description
ULFileTransfer constructor	Initializes a ULFileTransfer object. The connection must be opened before you can perform any operations against the database.

Public properties

Member name	Description
AuthenticationParms property	Specifies parameters for a custom user authentication script (MobiLink <code>authenticate_parameters</code> connection event).
AuthStatus property	Returns the authorization status code for the last file transfer attempt.
AuthValue property	Returns the return value from custom user authentication synchronization scripts.

Member name	Description
DestinationFileName property	Specifies the local file name for the downloaded file.
DestinationPath property	Specifies where to download the file.
DownloadedFile property	Checks whether the file was actually downloaded during the last file transfer attempt.
FileAuthCode property	Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.
FileName property	Specifies the name of the file to download.
ForceDownload property	Specifies whether to force the download of the file if it exists.
Password property	The MobiLink password for the user specified by UserName.
ResumePartialDownload property	Specifies whether to resume or discard a previous partial download.
Stream property	Specifies the MobiLink synchronization stream to use for the file transfer.
StreamErrorCode property	Returns the error reported by the stream itself for the last file transfer attempt.
StreamErrorSystem property	Returns the stream error system-specific code.
StreamParms property	Specifies the parameters to configure the synchronization stream.
UserName property	The user name that uniquely identifies the MobiLink client to the MobiLink server.
Version property	Specifies which synchronization script to use.

Public methods

Member name	Description
DownloadFile method	Download the file specified by the properties of this object.
DownloadFile method	Download the file specified by the properties of this object with progress events posted to the specified listener.

ULFileTransfer constructor

Initializes a ULFileTransfer object. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

Public Sub **New()**

C#

public **ULFileTransfer()**;

Remarks

You do not need a database connection to perform a file transfer, however, if your application will be using an UltraLite database with the UltraLite Engine runtime, you must set `ULDatabaseManager.RuntimeType` to the appropriate value before using this API or any other UltraLite.NET API.

The `ULFileTransfer` object needs to have the `ULFileTransfer.FileName`, `ULFileTransfer.Stream`, `ULFileTransfer.UserName`, and `ULFileTransfer.Version` set before it can transfer a file.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“DownloadFile method” on page 257](#)
- ◆ [“RuntimeType property” on page 179](#)
- ◆ [“FileName property” on page 251](#)
- ◆ [“Stream property” on page 254](#)
- ◆ [“UserName property” on page 256](#)
- ◆ [“Version property” on page 257](#)

AuthenticationParms property

Specifies parameters for a custom user authentication script (MobiLink `authenticate_parameters` connection event).

Syntax

Visual Basic

Public Property **AuthenticationParms** As String()

C#

public string [] **AuthenticationParms** {get;set;}

Property value

An array of strings, each containing an authentication parameter (null array entries result in a synchronization error). The default is a null reference (Nothing in Visual Basic), meaning no authentication parameters.

Remarks

Only the first 255 strings are used and each string should be no longer than 128 characters (longer strings are truncated when sent to the MobiLink server).

AuthStatus property

Returns the authorization status code for the last file transfer attempt.

Syntax

Visual Basic

Public Readonly Property **AuthStatus** As ULAuthStatusCode

C#

public ULAuthStatusCode **AuthStatus** {get;}

Property value

One of the ULAuthStatusCode values denoting the authorization status for the last file transfer attempt.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ULAuthStatusCode enumeration” on page 56](#)

AuthValue property

Returns the return value from custom user authentication synchronization scripts.

Syntax

Visual Basic

Public Readonly Property **AuthValue** As Long

C#

public long **AuthValue** {get;}

Property value

A long integer returned from custom user authentication synchronization scripts.

DestinationFileName property

Specifies the local file name for the downloaded file.

Syntax

Visual Basic

Public Property **DestinationFileName** As String

C#

```
public string DestinationFileName {get;set;}
```

Property value

A string specifying the local file name for the downloaded file. If the value is a null reference (Nothing in Visual Basic), FileName is used. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“FileName property” on page 251](#)

DestinationPath property

Specifies where to download the file.

Syntax**Visual Basic**

```
Public Property DestinationPath As String
```

C#

```
public string DestinationPath {get;set;}
```

Property value

A string specifying the destination directory of the file. The default is a null reference (Nothing in Visual Basic).

Remarks

The default destination directory varies depending on the device's operating system:

- ◆ For Windows Mobile devices, if the DestinationPath is a null reference (Nothing in Visual Basic), the file is stored in the root (\) directory.
- ◆ For desktop applications, if the DestinationPath is a null reference (Nothing in Visual Basic), the file is stored in the current directory.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ForceDownload property” on page 252](#)
- ◆ [“DownloadFile method” on page 257](#)

DownloadedFile property

Checks whether the file was actually downloaded during the last file transfer attempt.

Syntax

Visual Basic

Public Readonly Property **DownloadedFile** As Boolean

C#

```
public bool DownloadedFile {get;}
```

Property value

True if the file was downloaded, false otherwise.

Remarks

If the file is already up-to-date when the DownloadFile() is invoked, it will return true, but DownloadedFile will be false. If an error occurs and DownloadFile() returns false, DownloadedFile will be false.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“DownloadFile method” on page 257](#)
- ◆ [“DownloadFile method” on page 257](#)

FileAuthCode property

Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.

Syntax

Visual Basic

Public Readonly Property **FileAuthCode** As UInt16

C#

```
public ushort FileAuthCode {get;}
```

Property value

An unsigned short integer returned from the authenticate_file_transfer script for the last file transfer attempt.

FileName property

Specifies the name of the file to download.

Syntax

Visual Basic

Public Property **FileName** As String

C#

public string **FileName** {get;set;}

Property value

A string specifying the name of the file as recognized by the MobiLink server. This property has no default value, and must be explicitly set.

Remarks

FileName is the name of the file on the server running MobiLink. MobiLink will first search for this file in the UserName subdirectory and then in the root directory (the root directory is specified via the MobiLink server's -ftr option). FileName must not include any drive or path information, or the MobiLink server will be unable to find it. For example, "myfile.txt" is valid, but "somedir\myfile.txt", "..\myfile.txt", and "c:\myfile.txt" are all invalid.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“DestinationFileName property” on page 249](#)
- ◆ [“DownloadFile method” on page 257](#)

ForceDownload property

Specifies whether to force the download of the file if it exists.

Syntax

Visual Basic

Public Property **ForceDownload** As Boolean

C#

public bool **ForceDownload** {get;set;}

Property value

True to force the download of the file, false to perform normal download checks if file exists. The default is false.

Remarks

If ForceDownload is true, the file will always be downloaded, even if it hasn't changed, and any existing partial download is discarded. If ForceDownload is false, the file will only be downloaded if the server's version of the file is different from the client's. Note that if the file is changed on either the client or the server, specifying ForceDownload true will cause the server version to overwrite the client version.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ResumePartialDownload property” on page 253](#)
- ◆ [“DownloadFile method” on page 257](#)

Password property

The MobiLink password for the user specified by UserName.

Syntax**Visual Basic**

Public Property **Password** As String

C#

```
public string Password {get;set;}
```

Property value

A string specifying the MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning no password is specified.

Remarks

The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“UserName property” on page 256](#)
- ◆ [“DownloadFile method” on page 257](#)

ResumePartialDownload property

Specifies whether to resume or discard a previous partial download.

Syntax**Visual Basic**

Public Property **ResumePartialDownload** As Boolean

C#

```
public bool ResumePartialDownload {get;set;}
```

Property value

True to resume a previous partial download, false to discard a previous partial download. The default is false.

Remarks

UltraLite.NET has the ability to restart downloads that fail because of communication errors or user aborts through the `ULFileTransferListener`. UltraLite.NET processes the download as it is received. If a download is interrupted, then the partially download file is retained and can be resumed during the next file transfer.

If the file has been updated on the server, a partial download will be discarded and a new download started.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ForceDownload property” on page 252](#)
- ◆ [“DownloadFile method” on page 257](#)

Stream property

Specifies the MobiLink synchronization stream to use for the file transfer.

Syntax**Visual Basic**

Public Property **Stream** As ULStreamType

C#

```
public ULStreamType Stream {get;set;}
```

Property value

One of the `ULStreamType` values specifying the type of synchronization stream to use. The default is `ULStreamType.TCPIP`.

Remarks

Most synchronization streams require parameters to identify the MobiLink server address and control other behavior. These parameters are supplied by the `ULFileTransfer.StreamParms`.

If the stream type is set to a value that is invalid for the platform, the stream type is set to `ULStreamType.TCPIP`.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ULStreamType enumeration” on page 384](#)
- ◆ [“StreamParms property” on page 255](#)
- ◆ [“DownloadFile method” on page 257](#)
- ◆ [“ULStreamType enumeration” on page 384](#)

StreamErrorCode property

Returns the error reported by the stream itself for the last file transfer attempt.

Syntax

Visual Basic

Public Readonly Property **StreamErrorCode** As ULStreamErrorCode

C#

public ULStreamErrorCode **StreamErrorCode** {get;}

Property value

One of the ULStreamErrorCode values denoting the error reported by the stream itself, ULStreamErrorCode.NONE if no error occurred.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“ULStreamErrorCode enumeration” on page 366](#)
- ◆ [“ULStreamErrorCode enumeration” on page 366](#)

StreamErrorSystem property

Returns the stream error system-specific code.

Syntax

Visual Basic

Public Readonly Property **StreamErrorSystem** As Integer

C#

public int **StreamErrorSystem** {get;}

Property value

An integer denoting the stream error system-specific code.

StreamParms property

Specifies the parameters to configure the synchronization stream.

Syntax

Visual Basic

Public Property **StreamParms** As String

C#

```
public string StreamParms {get;set;}
```

Property value

A string, in the form of a semicolon-separated list of keyword-value pairs, specifying the parameters for the stream. The default is a null reference (Nothing in Visual Basic).

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams” \[MobiLink - Client Administration\]](#).

StreamParms is a string containing all the parameters used for synchronization streams. Parameters are specified as a semicolon-separated list of name=value pairs ("param1=value1;param2=value2").

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“Stream property” on page 254](#)
- ◆ [“DownloadFile method” on page 257](#)
- ◆ [“ULStreamType enumeration” on page 384](#)

UserName property

The user name that uniquely identifies the MobiLink client to the MobiLink server.

Syntax**Visual Basic**

```
Public Property UserName As String
```

C#

```
public string UserName {get;set;}
```

Property value

A string specifying the user name. This property has no default value, and must be explicitly set.

Remarks

The MobiLink server uses this value to locate the file to download. The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“Password property” on page 253](#)
- ◆ [“DownloadFile method” on page 257](#)

Version property

Specifies which synchronization script to use.

Syntax

Visual Basic

Public Property **Version** As String

C#

```
public string Version {get;set;}
```

Property value

A string specifying the version of the synchronization script to use. This property has no default value, and must be explicitly set.

Remarks

Each synchronization script in the consolidated database is marked with a version string. The version string allows an UltraLite application to choose from a set of synchronization scripts.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“DownloadFile method” on page 257](#)

DownloadFile method

Download the file specified by the properties of this object.

Syntax

Visual Basic

Overloads Public Function **DownloadFile()** As Boolean

C#

```
public bool DownloadFile();
```

Return value

True if successful, false otherwise (check ULFileTransfer.StreamErrorCode and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName is downloaded by the MobiLink server to the ULFileTransfer.DestinationPath using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version. Other properties that affect the download are

ULFileTransfer.DestinationFileName, ULFileTransfer.AuthenticationParms, ULFileTransfer.ForceDownload and ULFileTransfer.ResumePartialDownload.

To avoid file corruption, UltraLite.NET will download to a temporary file and only replace the destination file once the download has completed.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.DownloadedFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“DownloadFile method” on page 258](#)
- ◆ [“FileName property” on page 251](#)
- ◆ [“DestinationPath property” on page 250](#)
- ◆ [“Stream property” on page 254](#)
- ◆ [“UserName property” on page 256](#)
- ◆ [“Password property” on page 253](#)
- ◆ [“Version property” on page 257](#)
- ◆ [“DestinationFileName property” on page 249](#)
- ◆ [“AuthenticationParms property” on page 248](#)
- ◆ [“ForceDownload property” on page 252](#)
- ◆ [“ResumePartialDownload property” on page 253](#)
- ◆ [“AuthStatus property” on page 249](#)
- ◆ [“AuthValue property” on page 249](#)
- ◆ [“FileAuthCode property” on page 251](#)
- ◆ [“DownloadedFile property” on page 251](#)
- ◆ [“StreamErrorCode property” on page 255](#)
- ◆ [“StreamErrorSystem property” on page 255](#)
- ◆ [“StreamErrorCode property” on page 255](#)

DownloadFile method

Download the file specified by the properties of this object with progress events posted to the specified listener.

Syntax

Visual Basic

```
Overloads Public Function DownloadFile( _  
    ByVal listener As ULFileTransferProgressListener _  
) As Boolean
```

C#

```
public bool DownloadFile(  
    ULFileTransferProgressListener listener  
);
```

Parameters

- ◆ **listener** The object that receives file transfer progress events.

Return value

True if successful, false otherwise (check `ULFileTransfer.StreamErrorCode` and other status properties for reason).

Remarks

The file specified by the `ULFileTransfer.FileName` is downloaded by the MobiLink server to the `ULFileTransfer.DestinationPath` using the `ULFileTransfer.Stream`, `ULFileTransfer.UserName`, `ULFileTransfer.Password`, and `ULFileTransfer.Version`. Other properties that affect the download are `ULFileTransfer.DestinationFileName`, `ULFileTransfer.AuthenticationParms`, `ULFileTransfer.ForceDownload` and `ULFileTransfer.ResumePartialDownload`.

To avoid file corruption, UltraLite.NET will download to a temporary file and only replace the destination file once the download has completed.

A detailed result status is reported in this object's `ULFileTransfer.AuthStatus`, `ULFileTransfer.AuthValue`, `ULFileTransfer.FileAuthCode`, `ULFileTransfer.DownloadedFile`, `ULFileTransfer.StreamErrorCode`, and `ULFileTransfer.StreamErrorSystem`.

Errors may result in no data being sent to the listener.

See also

- ◆ [“ULFileTransfer class” on page 246](#)
- ◆ [“ULFileTransfer members” on page 246](#)
- ◆ [“FileName property” on page 251](#)
- ◆ [“DestinationPath property” on page 250](#)
- ◆ [“Stream property” on page 254](#)
- ◆ [“UserName property” on page 256](#)
- ◆ [“Password property” on page 253](#)
- ◆ [“Version property” on page 257](#)
- ◆ [“DestinationFileName property” on page 249](#)
- ◆ [“AuthenticationParms property” on page 248](#)
- ◆ [“ForceDownload property” on page 252](#)
- ◆ [“ResumePartialDownload property” on page 253](#)
- ◆ [“AuthStatus property” on page 249](#)
- ◆ [“AuthValue property” on page 249](#)
- ◆ [“FileAuthCode property” on page 251](#)
- ◆ [“DownloadedFile property” on page 251](#)
- ◆ [“StreamErrorCode property” on page 255](#)
- ◆ [“StreamErrorSystem property” on page 255](#)
- ◆ [“StreamErrorCode property” on page 255](#)

ULFileTransferProgressData class

UL Ext.: Returns file transfer progress monitoring data.

Syntax

Visual Basic

Public Class **ULFileTransferProgressData**

C#

public class **ULFileTransferProgressData**

See also

- ◆ [“ULFileTransferProgressData members” on page 260](#)
- ◆ [“ULFileTransferProgressListener interface” on page 263](#)

ULFileTransferProgressData members

Public static fields (shared)

Member name	Description
FLAG_IS_BLOCKING field	A flag indicating that the file transfer is blocked awaiting a response from the MobiLink server.

Public properties

Member name	Description
BytesReceived property	Returns the number of bytes received so far.
FileSize property	Returns the size of the file being transferred.
Flags property	Returns the current file transfer flags indicating additional information relating to the current state.
ResumedAtSize property	Returns the point in the file where the transfer was resumed.

FLAG_IS_BLOCKING field

A flag indicating that the file transfer is blocked awaiting a response from the MobiLink server.

Syntax

Visual Basic

Public Shared **FLAG_IS_BLOCKING** As Integer

C#

```
public const int FLAG_IS_BLOCKING;
```

BytesReceived property

Returns the number of bytes received so far.

Syntax

Visual Basic

Public Readonly Property **BytesReceived** As UInt64

C#

```
public ulong BytesReceived {get;}
```

Property value

The number of bytes received so far.

FileSize property

Returns the size of the file being transferred.

Syntax

Visual Basic

Public Readonly Property **FileSize** As UInt64

C#

```
public ulong FileSize {get;}
```

Property value

The size of the file in bytes.

Flags property

Returns the current file transfer flags indicating additional information relating to the current state.

Syntax

Visual Basic

Public Readonly Property **Flags** As Integer

C#

```
public int Flags {get;}
```

Property value

An integer containing a combination of flags or'ed together.

See also

- ◆ [“ULFileTransferProgressData class” on page 260](#)
- ◆ [“ULFileTransferProgressData members” on page 260](#)
- ◆ [“FLAG_IS_BLOCKING field” on page 260](#)

ResumedAtSize property

Returns the point in the file where the transfer was resumed.

Syntax

Visual Basic

Public Readonly Property **ResumedAtSize** As UInt64

C#

public ulong **ResumedAtSize** {get;}

Property value

The number of bytes transferred previously.

ULFileTransferProgressListener interface

UL Ext.: The listener interface for receiving file transfer progress events.

Syntax

Visual Basic

Public Interface **ULFileTransferProgressListener**

C#

public interface **ULFileTransferProgressListener**

See also

- ◆ [“ULFileTransferProgressListener members” on page 263](#)
- ◆ [“DownloadFile method” on page 258](#)

ULFileTransferProgressListener members

Public methods

Member name	Description
FileTransferProgressed method	Invoked during a file transfer to inform the user of progress. This method should return true to cancel the transfer or return false to continue.

FileTransferProgressed method

Invoked during a file transfer to inform the user of progress. This method should return true to cancel the transfer or return false to continue.

Syntax

Visual Basic

Public Function **FileTransferProgressed**(_
 ByVal *data* As ULFileTransferProgressData _
) As Boolean

C#

public bool **FileTransferProgressed**(
 ULFileTransferProgressData *data*
);

Parameters

- ◆ **data** A ULFileTransferProgressData object containing the latest file transfer progress data.

Return value

This method should return true to cancel the transfer or return false to continue.

Remarks

No UltraLite.NET API methods should be invoked during a FileTransferProgressed call.

See also

- ◆ [“ULFileTransferProgressListener interface” on page 263](#)
- ◆ [“ULFileTransferProgressListener members” on page 263](#)
- ◆ [“ULFileTransferProgressData class” on page 260](#)

ULIndexSchema class

UL Ext.: Represents the schema of an UltraLite table index.

Syntax

Visual Basic

NotInheritable Public Class **ULIndexSchema**

C#

public sealed class **ULIndexSchema**

Remarks

There is no constructor for this class. Index schemas are created using the `ULTableSchema.PrimaryKey`, `ULTableSchema.GetIndex(string)`, and `ULTableSchema.GetOptimalIndex(int)` of the `ULTableSchema`.

See also

- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“PrimaryKey property” on page 444](#)
- ◆ [“GetIndex method” on page 447](#)
- ◆ [“GetOptimalIndex method” on page 448](#)
- ◆ [“ULTableSchema class” on page 441](#)

ULIndexSchema members

Public properties

Member name	Description
ColumnCount property	Returns the number of columns in the index.
IsForeignKey property	Checks whether the index is a foreign key.
IsForeignKeyCheckOnCommit property	Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.
IsForeignKeyNullable property	Checks whether the foreign key is nullable.
IsOpen property	Determines whether the index schema is open or closed.
IsPrimaryKey property	Checks whether the index is the primary key.
IsUniqueIndex property	Checks whether the index is unique.
IsUniqueKey property	Checks whether the index is a unique key.
Name property	Returns the name of the index.

Member name	Description
ReferencedIndexName property	The name of the referenced primary index if the index is a foreign key.
ReferencedTableName property	The name of the referenced primary table if the index is a foreign key.

Public methods

Member name	Description
GetColumnName method	Returns the name of the <i>colOrdinalInIndex</i> 'th column in this index.
IsColumnDescending method	Checks whether the named column is used in descending order by the index.

ColumnCount property

Returns the number of columns in the index.

Syntax

Visual Basic

Public Readonly Property **ColumnCount** As Short

C#

```
public short ColumnCount {get;}
```

Property value

The number of columns in the index.

Remarks

Column ordinals in indexes range from 1 to ColumnCount, inclusive.

Column ordinals and count may change during a schema upgrade. Column ordinals from an index are different than the column IDs in a table or another index, even if they refer to the same physical column in a particular table.

IsForeignKey property

Checks whether the index is a foreign key.

Syntax

Visual Basic

Public Readonly Property **IsForeignKey** As Boolean

C#

```
public bool IsForeignKey {get;}
```

Property value

True if the index is the foreign key, false if the index is not the foreign key.

Remarks

Columns in a foreign key may reference another table's non-null, unique index.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

IsForeignKeyCheckOnCommit property

Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.

Syntax**Visual Basic**

Public Readonly Property **IsForeignKeyCheckOnCommit** As Boolean

C#

```
public bool IsForeignKeyCheckOnCommit {get;}
```

Property value

True if referential integrity is checked on commits, false if it is checked on inserts and updates.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred (including index is not a foreign key).

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“IsForeignKey property” on page 266](#)

IsForeignKeyNullable property

Checks whether the foreign key is nullable.

Syntax**Visual Basic**

Public Readonly Property **IsForeignKeyNullable** As Boolean

C#

```
public bool IsForeignKeyNullable {get;}
```

Property value

True if the foreign key is nullable, false if the foreign key is not nullable.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred (including index is not a foreign key).

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“IsForeignKey property” on page 266](#)

IsOpen property

Determines whether the index schema is open or closed.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen {get;}
```

Property value

True if the index schema is open, otherwise false.

IsPrimaryKey property

Checks whether the index is the primary key.

Syntax

Visual Basic

Public Readonly Property **IsPrimaryKey** As Boolean

C#

```
public bool IsPrimaryKey {get;}
```

Property value

True if the index is the primary key, false if the index is not the primary key.

Remarks

Columns in the primary key may not be null.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

IsUniqueIndex property

Checks whether the index is unique.

Syntax**Visual Basic**

Public Readonly Property **IsUniqueIndex** As Boolean

C#

```
public bool IsUniqueIndex {get;}
```

Property value

True if the index is unique, false if the index is not unique.

Remarks

Columns in a unique index may be null.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

IsUniqueKey property

Checks whether the index is a unique key.

Syntax**Visual Basic**

Public Readonly Property **IsUniqueKey** As Boolean

C#

```
public bool IsUniqueKey {get;}
```

Property value

True if the index is a unique key, false if the index is not a unique key.

Remarks

Columns in a unique key may not be null.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Name property

Returns the name of the index.

Syntax

Visual Basic

Public Readonly Property **Name** As String

C#

public string **Name** {get;}

Property value

A string specifying the name of the index.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

ReferencedIndexName property

The name of the referenced primary index if the index is a foreign key.

Syntax

Visual Basic

Public Readonly Property **ReferencedIndexName** As String

C#

public string **ReferencedIndexName** {get;}

Property value

A string specifying the name of the referenced primary index.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred (including index is not a foreign key).

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“IsForeignKey property” on page 266](#)

ReferencedTableName property

The name of the referenced primary table if the index is a foreign key.

Syntax

Visual Basic

```
Public Readonly Property ReferencedTableName As String
```

C#

```
public string ReferencedTableName {get;}
```

Property value

A string specifying the name of the referenced primary table.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred (including index is not a foreign key).

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“IsForeignKey property” on page 266](#)

GetColumnName method

Returns the name of the *colOrdinalInIndex*'th column in this index.

Syntax

Visual Basic

```
Public Function GetColumnName( _  
    ByVal colOrdinalInIndex As Short _  
) As String
```

C#

```
public string GetColumnName(  
    short colOrdinalInIndex  
);
```

Parameters

- ◆ **colOrdinalInIndex** The ordinal of the desired column in the index. The value must be in the range [1,ColumnCount].

Return value

The name of the column.

Remarks

Column ordinals and count may change during a schema upgrade. Column ordinals from an index are different than the column IDs in a table or another index, even if they refer to the same physical column in a particular table.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“ColumnCount property” on page 266](#)
- ◆ [“ColumnCount property” on page 266](#)

IsColumnDescending method

Checks whether the named column is used in descending order by the index.

Syntax**Visual Basic**

```
Public Function IsColumnDescending( _  
    ByVal name As String _  
) As Boolean
```

C#

```
public bool IsColumnDescending(  
    string name  
);
```

Parameters

- ◆ **name** The name of the column.

Return value

True if the column is used in descending order, false if the column is used in ascending order.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULIndexSchema class” on page 265](#)
- ◆ [“ULIndexSchema members” on page 265](#)
- ◆ [“GetColumnName method” on page 271](#)
- ◆ [“ColumnCount property” on page 266](#)

ULInfoMessageEventArgs class

Provides data for the ULConnection.InfoMessage event.

Syntax

Visual Basic

NotInheritable Public Class **ULInfoMessageEventArgs**
Inherits EventArgs

C#

```
public sealed class ULInfoMessageEventArgs :  
EventArgs
```

See also

- ◆ [“ULInfoMessageEventArgs members” on page 273](#)
- ◆ [“InfoMessage event” on page 119](#)

ULInfoMessageEventArgs members

Public properties

Member name	Description
Message property	The informational or warning message string returned by the database.
NativeError property	The SQL code corresponding to the informational message or warning returned by the database.
Source property	The name of the ADO.NET data provider returning the message.

Public methods

Member name	Description
ToString method	Returns the string representation of the ULConnection.InfoMessage event.

Message property

The informational or warning message string returned by the database.

Syntax

Visual Basic

Public Readonly Property **Message** As String

C#

```
public string Message {get;}
```

Property value

A string containing the informational or warning message.

NativeError property

The SQL code corresponding to the informational message or warning returned by the database.

Syntax

Visual Basic

Public Readonly Property **NativeError** As ULSQLCode

C#

```
public ULSQLCode NativeError {get;}
```

Property value

An informational or warning ULSQLCode value.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 273](#)
- ◆ [“ULInfoMessageEventArgs members” on page 273](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)

Source property

The name of the ADO.NET data provider returning the message.

Syntax

Visual Basic

Public Readonly Property **Source** As String

C#

```
public string Source {get;}
```

Property value

The string "UltraLite.NET Data Provider".

ToString method

Returns the string representation of the ULConnection.InfoMessage event.

Syntax**Visual Basic**

Overrides Public Function **ToString()** As String

C#

public override string **ToString()**;

Property value

The string "UltraLite.NET Data Provider".

Return value

The informational or warning message string.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 273](#)
- ◆ [“ULInfoMessageEventArgs members” on page 273](#)
- ◆ [“InfoMessage event” on page 119](#)

ULInfoMessageEventHandler delegate

Represents the method that will handle the `ULConnection.InfoMessage` event.

Syntax

Visual Basic

```
Public Delegate Sub ULInfoMessageEventHandler( _  
    ByVal obj As Object, _  
    ByVal args As ULInfoMessageEventArgs _  
)
```

C#

```
public delegate void ULInfoMessageEventHandler(  
    object obj,  
    ULInfoMessageEventArgs args  
);
```

Parameters

- ◆ **obj** The connection sending the event.
- ◆ **args** The `ULInfoMessageEventArgs` object that contains the event data.

See also

- ◆ [“InfoMessage event” on page 119](#)
- ◆ [“ULInfoMessageEventArgs class” on page 273](#)

ULParameter class

Represents a parameter to a ULCommand.

Syntax

Visual Basic

NotInheritable Public Class **ULParameter**
Inherits MarshalByRefObject

C#

public sealed class **ULParameter** :
MarshalByRefObject

Remarks

A ULParameter object can be created directly using one of its many constructors, or using the ULCommand.CreateParameter method. Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using the ULParameter(string,object) constructor. For example:

```
' Visual Basic
Dim p As ULParameter = New ULParameter( "", CType( 0, Object ) )

// C#
ULParameter p = new ULParameter( "", (object)0 );
```

Parameters (including those created by ULCommand.CreateParameter) must be added to a ULCommand.Parameters collection to be used. All parameters are treated as positional parameters and are used by a command in the order that they were added.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Implements: System.Data.IDbDataParameter, System.Data.IDataParameter

See also

- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“CreateParameter method” on page 70](#)
- ◆ [“ULParameter constructor” on page 280](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [“Value property” on page 292](#)
- ◆ [IDbDataParameter](#)
- ◆ [IDataParameter](#)

ULParameter members

Public constructors

Member name	Description
ULParameter constructor	Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.
ULParameter constructor	Initializes a ULParameter object with the specified parameter name and value.
ULParameter constructor	Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.
ULParameter constructor	Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.
ULParameter constructor	Initializes a ULParameter object with the specified parameter name, data type, and length. This constructor is not recommended; it is provided for compatibility with other data providers.
ULParameter constructor	Initializes a ULParameter object with the specified parameter name, data type, length, direction, nullability, numeric precision, numeric scale, source column, source version, and value. This constructor is not recommended; it is provided for compatibility with other data providers.

Public properties

Member name	Description
DbType property	Specifies the System.Data.DbType of the parameter
Direction property	A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.
IsNullable property	Specifies whether the parameter accepts null values.
Offset property	Specifies the offset to the ULParameter.Value.
ParameterName property	Specifies the name of the parameter.
Precision property	Specifies the maximum number of digits used to represent the ULParameter.Value.
Scale property	Specifies the number of decimal places to which ULParameter.Value is resolved.
Size property	Specifies the maximum size of the data within the column.
SourceColumn property	Specifies the name of the source column mapped to the DataSet and used for loading or returning the value.

Member name	Description
SourceVersion property	The System.Data.DataRowVersion to use when loading ULParameter.Value.
ULDbType property	Specifies the iAnywhere.Data.UltraLite.ULDbType of the parameter
Value property	Specifies the value of the parameter.

Public methods

Member name	Description
ToString method	Returns the string representation of this instance.

ULParameter constructor

Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

Syntax

Visual Basic

Overloads Public Sub **New()**

C#

public **ULParameter()**;

Example

The following code creates a ULParameter with the value 3 and adds it to a ULCommand called cmd.

```
' Visual Basic
Dim p As ULParameter = New ULParameter
p.Value = 3
cmd.Parameters.Add( p )

// C#
ULParameter p = new ULParameter();
p.Value = 3;
cmd.Parameters.Add( p );
```

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)
- ◆ [“ULParameter constructor” on page 280](#)
- ◆ [“ULCommand class” on page 57](#)

ULParameter constructor

Initializes a ULParameter object with the specified parameter name and value.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal value As Object _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    object value  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **value** A System.Object that is to be the value of the parameter.

Remarks

Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using this constructor.

Example

The following code creates a ULParameter with the value 0 and adds it to a ULCommand called cmd.

```
' Visual Basic  
cmd.Parameters.Add( New ULParameter( "", CType( 0, Object ) ) )  
  
// C#  
cmd.Parameters.Add( new ULParameter( "", (object)0 ) );
```

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULParameter constructor” on page 279](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [Object](#)

ULParameter constructor

Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULParameter constructor” on page 279](#)
- ◆ [“ULParameter constructor” on page 280](#)
- ◆ [“Value property” on page 292](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULDbType enumeration” on page 238](#)

ULParameter constructor

Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType, _  
    ByVal size As Integer _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType,  
    int size  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULParameter constructor” on page 279](#)
- ◆ [“ULParameter constructor” on page 280](#)
- ◆ [“Value property” on page 292](#)
- ◆ [“ULDbType enumeration” on page 238](#)

ULParameter constructor

Initializes a ULParameter object with the specified parameter name, data type, and length. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax**Visual Basic**

```
Overloads Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType, _  
    ByVal size As Integer, _  
    ByVal sourceColumn As String _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType,  
    int size,
```

```
string sourceColumn  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.
- ◆ **sourceColumn** The name of the source column to map.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULParameter constructor” on page 279](#)
- ◆ [“ULParameter constructor” on page 280](#)
- ◆ [“Value property” on page 292](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULCommand class” on page 57](#)

ULParameter constructor

Initializes a ULParameter object with the specified parameter name, data type, length, direction, nullability, numeric precision, numeric scale, source column, source version, and value. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax

Visual Basic

```
Overloads Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType, _  
    ByVal size As Integer, _  
    ByVal direction As ParameterDirection, _  
    ByVal isNullable As Boolean, _  
    ByVal precision As Byte, _  
    ByVal scale As Byte, _  
    ByVal sourceColumn As String, _  
    ByVal sourceVersion As DataRowVersion, _  
    ByVal value As Object _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType,  
    int size,  
    ParameterDirection direction,  
    bool isNullable,  
    byte precision,  
    byte scale,  
    string sourceColumn,  
    DataRowVersion sourceVersion,  
    object value  
);
```

Parameters

- ♦ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ♦ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ♦ **size** The length of the parameter.
- ♦ **direction** One of the System.Data.ParameterDirection values.
- ♦ **isNullable** True if the value of the field can be null; otherwise, false.
- ♦ **precision** The total number of digits to the left and right of the decimal point to which Value is resolved.
- ♦ **scale** The total number of decimal places to which Value is resolved.
- ♦ **sourceColumn** The name of the source column to map.
- ♦ **sourceVersion** One of the System.Data.DataRowVersion values.
- ♦ **value** An System.Object that is to be the value of the parameter.

Exceptions

- ♦ [ULException class](#) - Only the System.Data.ParameterDirection.Input direction is supported in UltraLite.NET.

See also

- ♦ [“ULParameter class” on page 277](#)
- ♦ [“ULParameter members” on page 278](#)
- ♦ [“ULParameter constructor” on page 279](#)
- ♦ [“ULParameter constructor” on page 280](#)
- ♦ [“ULDbType enumeration” on page 238](#)
- ♦ [ParameterDirection](#)
- ♦ [DataRowVersion](#)
- ♦ [Object](#)
- ♦ [ParameterDirection.Input](#)

- ◆ [“ULCommand class” on page 57](#)

DbType property

Specifies the System.Data.DbType of the parameter

Syntax

Visual Basic

```
NotOverridable Public Property DbType As DbType _  
    Implements IDataParameter.DbType
```

C#

```
public DbType DbType {get;set;}
```

Property value

One of the System.Data.DbType values.

Remarks

The ULParameter.ULDbType and DbType properties are linked. Therefore, setting the DbType changes the ULParameter.ULDbType to a supporting iAnywhere.Data.UltraLite.ULDbType.

Exceptions

- ◆ [ArgumentException](#) - There is no mapping from the specified value to a iAnywhere.Data.UltraLite.ULDbType, hence, the specified value is not supported.

Implements

[IDataParameter.DbType](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [DbType](#)
- ◆ [DbType](#)
- ◆ [“ULDbType property” on page 291](#)
- ◆ [“ULDbType enumeration” on page 238](#)

Direction property

A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.

Syntax

Visual Basic

NotOverridable Public Property **Direction** As ParameterDirection _
Implements IDataParameter.Direction

C#

public ParameterDirection **Direction** {get;set;}

Property value

One of the System.Data.ParameterDirection values.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Exceptions

- ◆ [ULException class](#) - Only the System.Data.ParameterDirection.Input direction is supported in UltraLite.NET.

Implements

[IDataParameter.Direction](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [ParameterDirection](#)
- ◆ [ParameterDirection.Input](#)
- ◆ [“Value property” on page 292](#)

IsNullable property

Specifies whether the parameter accepts null values.

Syntax

Visual Basic

Public Property **IsNullable** As Boolean _
Implements IDataParameter.IsNullable

C#

public bool **IsNullable** {get;set;}

Property value

True if null values are accepted, false otherwise. The default is false. Null values are handled using the DBNull class.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Implements

[IDataParameter.IsNullable](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

Offset property

Specifies the offset to the ULParameter.Value.

Syntax

Visual Basic

Public Property **Offset** As Integer

C#

```
public int Offset {get;set;}
```

Property value

The offset to the value. The default is 0.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

ParameterName property

Specifies the name of the parameter.

Syntax

Visual Basic

NotOverridable Public Property **ParameterName** As String _
Implements IDataParameter.ParameterName

C#

```
public string ParameterName {get;set;}
```

Property value

A string representing the name of the parameter, or an empty string (""), for unnamed parameters. Specifying a null reference (Nothing in Visual Basic) results in an empty string being used.

Remarks

In UltraLite.NET, parameter names are not used by ULCommand. All parameters are treated as positional parameters and are used by a command in the order that they were added.

Implements

[IDataParameter.ParameterName](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULCommand class” on page 57](#)

Precision property

Specifies the maximum number of digits used to represent the ULParameter.Value.

Syntax**Visual Basic**

```
NotOverridable Public Property Precision As Byte _  
    Implements IDbDataParameter.Precision
```

C#

```
public byte Precision {get;set;}
```

Property value

The maximum number of digits used to represent the ULParameter.Value. The default value is 0, which indicates that the data provider sets the precision for the ULParameter.Value.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Exceptions

- ◆ [ArgumentException](#) - The value is greater than 38.

Implements

[IDbDataParameter.Precision](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

Scale property

Specifies the number of decimal places to which ULParameter.Value is resolved.

Syntax**Visual Basic**

```
NotOverridable Public Property Scale As Byte _  
    Implements IDbDataParameter.Scale
```

C#

```
public byte Scale {get;set;}
```

Property value

The number of decimal places to which ULParameter.Value is resolved. The default is 0.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Implements

[IDbDataParameter.Scale](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

Size property

Specifies the maximum size of the data within the column.

Syntax**Visual Basic**

```
NotOverridable Public Property Size As Integer _  
    Implements IDbDataParameter.Size
```

C#

```
public int Size {get;set;}
```

Property value

The maximum size of the data within the column. The default value is inferred from the parameter value. The Size property is used for binary and string types.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Implements

[IDbDataParameter.Size](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

SourceColumn property

Specifies the name of the source column mapped to the DataSet and used for loading or returning the value.

Syntax**Visual Basic**

```
NotOverridable Public Property SourceColumn As String _  
    Implements IDataParameter.SourceColumn
```

C#

```
public string SourceColumn {get;set;}
```

Property value

A string specifying the name of the source column mapped to the DataSet and used for loading or returning the value.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Implements

[IDataParameter.SourceColumn](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)

SourceVersion property

The System.Data.DataRowVersion to use when loading ULParameter.Value.

Syntax

Visual Basic

```
NotOverridable Public Property SourceVersion As DataRowVersion _  
    Implements IDataParameter.SourceVersion
```

C#

```
public DataRowVersion SourceVersion {get;set;}
```

Implements

[IDataParameter.SourceVersion](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“Value property” on page 292](#)
- ◆ [DataRowVersion](#)

ULDbType property

Specifies the iAnywhere.Data.UltraLite.ULDbType of the parameter

Syntax

Visual Basic

```
Public Property ULDbType As ULDbType
```

C#

```
public ULDbType ULDbType {get;set;}
```

Property value

One of the iAnywhere.Data.UltraLite.ULDbType values.

Remarks

The ULDbType and ULParameter.DbType are linked. Therefore, setting the ULDbType changes the ULParameter.DbType to a supporting System.Data.DbType.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)

- ◆ [“Value property” on page 292](#)
- ◆ [“DbType property” on page 285](#)
- ◆ [DbType](#)
- ◆ [“ULDbType enumeration” on page 238](#)

Value property

Specifies the value of the parameter.

Syntax

Visual Basic

NotOverridable Public Property **Value** As Object _
Implements IDataParameter.Value

C#

public object **Value** {get;set;}

Property value

A System.Object that specifies the value of the parameter.

Remarks

The value is sent as-is to the data provider without any type conversion or mapping. When the command is executed, the command attempts to convert the value to the required type, signaling a `ULException` with `ULSQLCode.SQLE_CONVERSION_ERROR` if it cannot convert the value.

Implements

[IDataParameter.Value](#)

See also

- ◆ [“ULParameter class” on page 277](#)
- ◆ [“ULParameter members” on page 278](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [Object](#)

ToString method

Returns the string representation of this instance.

Syntax

Visual Basic

Overrides Public Function **ToString()** As String

C#

```
public override string ToString();
```

Return value

The name of the parameter.

ULParameterCollection class

Represents all parameters to a ULCommand.

Syntax

Visual Basic

NotInheritable Public Class **ULParameterCollection**

C#

public sealed class **ULParameterCollection**

Remarks

All parameters in the collection are treated as positional parameters and are specified in the same order as the question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

There is no constructor for ULParameterCollection. You obtain a ULParameterCollection from the ULCommand.Parameters.

Implements: System.Data.IDataParameterCollection

See also

- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“Parameters property” on page 67](#)
- ◆ [IDataParameterCollection](#)

ULParameterCollection members

Public properties

Member name	Description
Count property	Returns the number of ULParameter objects in the collection.
Item property	Returns the ULParameter at the specified index. In C#, this property is the indexer for the ULParameterCollection class.
Item property	Returns the ULParameter with the specified name. In C#, this property is the indexer for the ULParameterCollection class.

Public methods

Member name	Description
Add method	Adds a ULParameter to the collection.
Add method	Adds a ULParameter to the collection.
Add method	Adds a new ULParameter, created using the specified parameter name and value, to the collection.
Add method	Adds a new ULParameter, created using the specified parameter name and data type, to the collection.
Add method	Adds a new ULParameter, created using the specified parameter name, data type, and length, to the collection.
Add method	Adds a new ULParameter, created using the specified parameter name, data type, length, and source column name, to the collection.
Clear method	Removes all the parameters from the collection.
Contains method	Checks whether a ULParameter exists in the collection.
Contains method	Checks whether a ULParameter with the specified name exists in the collection.
CopyTo method	Copies ULParameter objects from the ULParameterCollection to the specified array.
GetEnumerator method	Returns an enumerator for the collection.
IndexOf method	Returns the location of the ULParameter in the collection.
IndexOf method	Returns the location of the ULParameter with the specified name in the collection.
Insert method	Inserts an ULParameter in the collection at the specified index.
Remove method	Removes an ULParameter from the collection.
RemoveAt method	Removes the parameter at the specified index in the collection.
RemoveAt method	Removes the parameter with the specified name from the collection.

Count property

Returns the number of ULParameter objects in the collection.

Syntax**Visual Basic**

NotOverridable Public Readonly Property **Count** As Integer _
Implements ICollection.Count

C#

```
public int Count {get;}
```

Property value

The number of ULParameter objects in the collection.

Implements

[ICollection.Count](#)

Item property

Returns the ULParameter at the specified index. In C#, this property is the indexer for the ULParameterCollection class.

Syntax

Visual Basic

Public Property **Item** As ULParameter

C#

```
public ULParameter Item {get;set;}
```

Parameters

- ♦ **index** The zero-based index of the parameter to retrieve. The value must be in the range [0,ULParameterCollection.Count-1]. The first parameter in the collection has an index value of zero.

Return value

The ULParameter at the specified index.

Exceptions

- ♦ [System.IndexOutOfRangeException](#) - The index is invalid.

See also

- ♦ [“ULParameterCollection class” on page 294](#)
- ♦ [“ULParameterCollection members” on page 294](#)
- ♦ [“ULParameter class” on page 277](#)

Item property

Returns the ULParameter with the specified name. In C#, this property is the indexer for the ULParameterCollection class.

Syntax

Visual Basic

Public Property **Item** As ULParameter

C#

public ULParameter **Item** {get;set;}

Parameters

- ◆ **parameterName** The name of the parameter to retrieve.

Return value

The ULParameter with the specified name.

Exceptions

- ◆ [System.ArgumentNullException](#) - You cannot set a parameter using a null (Nothing in Visual Basic) parameter name.
- ◆ [System.IndexOutOfRangeException](#) - There is no parameter with the specified name.

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Item property” on page 201](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetValue method” on page 229](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“ULParameter class” on page 277](#)

Add method

Adds a ULParameter to the collection.

Syntax

Visual Basic

Overloads NotOverridable Public Function **Add**(_
 ByVal *value* As Object _
) As Integer _
 Implements IList.Add

C#

public int **Add**(
 object *value*
);

Parameters

- ◆ **value** The ULParameter object to add to the collection.

Return value

The index of the new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

Exceptions

- ◆ [ArgumentException](#) - The ULParameter object can only be added to the collection once.
- ◆ [System.ArgumentNullException](#) - The value cannot be null (Nothing in Visual Basic).
- ◆ [System.InvalidCastException](#) - The value specified must be a ULParameter.

Implements

[IList.Add](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 298](#)
- ◆ [“Add method” on page 299](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ULParameter class” on page 277](#)

Add method

Adds a ULParameter to the collection.

Syntax

Visual Basic

```
Overloads Public Function Add( _  
    ByVal value As ULParameter _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    ULParameter value  
);
```

Parameters

- ◆ **value** The ULParameter object to add to the collection.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

Exceptions

- ◆ [ArgumentException](#) - The ULParameter object can only be added to the collection once.
- ◆ [System.ArgumentNullException](#) - The value cannot be null (Nothing in Visual Basic).

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 299](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [“CommandText property” on page 63](#)

Add method

Adds a new ULParameter, created using the specified parameter name and value, to the collection.

Syntax

Visual Basic

```
Overloads Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal value As Object _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    object value  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **value** A System.Object that is to be the value of the parameter.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using this method.

Example

The following code adds a ULParameter with the value 0 to a ULCommand called cmd.

```
' Visual Basic
cmd.Parameters.Add( "", CType( 0, Object ) )

// C#
cmd.Parameters.Add( "", (object)0 );
```

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 298](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [Object](#)

Add method

Adds a new ULParameter, created using the specified parameter name and data type, to the collection.

Syntax

Visual Basic

Overloads Public Function **Add**(_
 ByVal *parameterName* As String, _

```
    ByVal ulDbType As ULDbType _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    ULDbType ulDbType  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 298](#)
- ◆ [“Add method” on page 299](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULDbType enumeration” on page 238](#)

Add method

Adds a new ULParameter, created using the specified parameter name, data type, and length, to the collection.

Syntax**Visual Basic**

Overloads Public Function **Add**(_
 ByVal parameterName As String, _
 ByVal ulDbType As ULDbType, _

```
    ByVal size As Integer _  
  ) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    ULDbType ulDbType,  
    int size  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 298](#)
- ◆ [“Add method” on page 299](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULDbType enumeration” on page 238](#)

Add method

Adds a new ULParameter, created using the specified parameter name, data type, length, and source column name, to the collection.

Syntax**Visual Basic**


```
Overloads Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal ulDbType As ULDbType, _  
    ByVal size As Integer, _  
    ByVal sourceColumn As String _  
    ) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    ULDbType ulDbType,  
    int size,  
    string sourceColumn  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.
- ◆ **sourceColumn** The name of the source column to map.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Add method” on page 298](#)
- ◆ [“Add method” on page 299](#)
- ◆ [“ULParameter class” on page 277](#)
- ◆ [“CommandText property” on page 63](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULDbType enumeration” on page 238](#)

Clear method

Removes all the parameters from the collection.

Syntax

Visual Basic

```
NotOverridable Public Sub Clear() _  
    Implements IList.Clear
```

C#

```
public void Clear();
```

Implements

[IList.Clear](#)

Contains method

Checks whether a ULParameter exists in the collection.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function Contains( _  
    ByVal value As Object _  
) As Boolean _  
    Implements IList.Contains
```

C#

```
public bool Contains(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to check for.

Return value

True if the collection contains the ULParameter, false otherwise.

Implements

[IList.Contains](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Contains method” on page 305](#)
- ◆ [“ULParameter class” on page 277](#)

Contains method

Checks whether a ULParameter with the specified name exists in the collection.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function Contains( _  
    ByVal value As String _  
) As Boolean _  
    Implements IDataParameterCollection.Contains
```

C#

```
public bool Contains(  
    string value  
);
```

Parameters

- ◆ **value** The name of the parameter to search for.

Return value

True if the collection contains the ULParameter, false otherwise.

Implements

[IDataParameterCollection.Contains](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“Contains method” on page 304](#)
- ◆ [“ULParameter class” on page 277](#)

CopyTo method

Copies ULParameter objects from the ULParameterCollection to the specified array.

Syntax

Visual Basic

```
NotOverridable Public Sub CopyTo( _  
    ByVal array As System.Array, _  
    ByVal index As Integer _  
) _  
    Implements ICollection.CopyTo
```

C#

```
public void CopyTo(  
    System.Array array,
```

```
    int index  
);
```

Parameters

- ♦ **array** The array into which to copy the ULParameter objects.
- ♦ **index** The starting index of the array.

Implements

[ICollection.CopyTo](#)

See also

- ♦ [“ULParameterCollection class” on page 294](#)
- ♦ [“ULParameterCollection members” on page 294](#)
- ♦ [“ULParameter class” on page 277](#)

GetEnumerator method

Returns an enumerator for the collection.

Syntax

Visual Basic

```
NotOverridable Public Function GetEnumerator() As System.Collections.IEnumerator _  
    Implements IEnumerable.GetEnumerator
```

C#

```
public System.Collections.IEnumerator GetEnumerator();
```

Return value

An ArrayList enumerator enumerating the parameters in the collection.

Implements

[IEnumerable.GetEnumerator](#)

IndexOf method

Returns the location of the ULParameter in the collection.

Syntax

Visual Basic

```
Overloads NotOverridable Public Function IndexOf( _  
    ByVal value As Object _  
    ) As Integer _  
    Implements IList.IndexOf
```

C#

```
public int IndexOf(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to locate.

Return value

The zero-based index of the ULParameter in the collection or -1 if the parameter is not found.

Exceptions

- ◆ [System.InvalidCastException](#) - The value specified must be a ULParameter.

Implements

[IList.IndexOf](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“IndexOf method” on page 307](#)
- ◆ [“ULParameter class” on page 277](#)

IndexOf method

Returns the location of the ULParameter with the specified name in the collection.

Syntax**Visual Basic**

```
Overloads NotOverridable Public Function IndexOf(  
    ByVal parameterName As String _  
) As Integer _  
    Implements IDataParameterCollection.IndexOf
```

C#

```
public int IndexOf(  
    string parameterName  
);
```

Parameters

- ◆ **parameterName** The name of the parameter to locate.

Return value

The zero-based index of the ULParameter in the collection or -1 if the parameter is not found.

Implements

[IDataParameterCollection.IndexOf](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“IndexOf method” on page 306](#)
- ◆ [“ULParameter class” on page 277](#)

Insert method

Inserts an ULParameter in the collection at the specified index.

Syntax

Visual Basic

```
NotOverridable Public Sub Insert( _  
    ByVal index As Integer, _  
    ByVal value As Object _  
) _  
    Implements IList.Insert
```

C#

```
public void Insert(  
    int index,  
    object value  
);
```

Parameters

- ◆ **index** The zero-based index where the parameter is to be inserted within the collection.
- ◆ **value** The ULParameter object to insert.

Exceptions

- ◆ [System.ArgumentNullException](#) - You cannot set a parameter using a null reference (Nothing in Visual Basic).
- ◆ [System.IndexOutOfRangeException](#) - The index is invalid.
- ◆ [System.InvalidCastException](#) - The value specified must be a ULParameter.

Implements

[IList.Insert](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“ULParameter class” on page 277](#)

Remove method

Removes an ULParameter from the collection.

Syntax

Visual Basic

```
NotOverridable Public Sub Remove( _  
    ByVal value As Object _  
) _  
    Implements IList.Remove
```

C#

```
public void Remove(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to remove.

Exceptions

- ◆ [ArgumentException](#) - The collection does not contain the specified parameter.
- ◆ [System.ArgumentNullException](#) - You cannot set a parameter using a null reference (Nothing in Visual Basic).
- ◆ [System.InvalidCastException](#) - The value specified must be a ULParameter.

Implements

[IList.Remove](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“ULParameter class” on page 277](#)

RemoveAt method

Removes the parameter at the specified index in the collection.

Syntax

Visual Basic

```
Overloads NotOverridable Public Sub RemoveAt( _  
    ByVal index As Integer _  
) _  
    Implements IList.RemoveAt
```

C#

```
public void RemoveAt(  
    int index  
);
```

Parameters

- ◆ **index** The zero-based index of the parameter to remove. The value must be in the range [0,ULParameterCollection.Count-1]. The first parameter in the collection has an index value of zero.

Exceptions

- ◆ [System.IndexOutOfRangeException](#) - The index is invalid.

Implements[IList.RemoveAt](#)**See also**

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“RemoveAt method” on page 310](#)
- ◆ [“Count property” on page 295](#)

RemoveAt method

Removes the parameter with the specified name from the collection.

Syntax**Visual Basic**

```
Overloads NotOverridable Public Sub RemoveAt( _  
    ByVal parameterName As String _  
) _  
    Implements IDataParameterCollection.RemoveAt
```

C#

```
public void RemoveAt(  
    string parameterName  
);
```

Parameters

- ◆ **parameterName** The name of the parameter to retrieve.

Exceptions

- ◆ [System.IndexOutOfRangeException](#) - There is no parameter with the specified name.

Implements[IDataParameterCollection.RemoveAt](#)

See also

- ◆ [“ULParameterCollection class” on page 294](#)
- ◆ [“ULParameterCollection members” on page 294](#)
- ◆ [“RemoveAt method” on page 309](#)

ULPublicationSchema class

UL Ext.: Represents the schema of an UltraLite publication.

Syntax

Visual Basic

NotInheritable Public Class **ULPublicationSchema**

C#

public sealed class **ULPublicationSchema**

Remarks

There is no constructor for this class. Publication schemas are created using the `ULDatabaseSchema.GetPublicationSchema` method of the `ULDatabaseSchema` class.

UltraLite methods requiring a publication mask actually require a set of publications to check. A set is formed by or'ing the publication masks of individual publications. For example:

```
' Visual Basic
Dim mask As Integer = pub1.Mask Or pub2.Mask

// C#
int mask = pub1.Mask | pub2.Mask;
```

Two special mask values are also provided by this class. `ULPublicationSchema.SYNC_ALL_DB` corresponds to the entire database. `ULPublicationSchema.SYNC_ALL_PUBS` corresponds to all publications.

Note: Publication masks may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached masks after a schema upgrade.

See also

- ◆ [“ULPublicationSchema members” on page 313](#)
- ◆ [“Mask property” on page 314](#)
- ◆ [“GetLastDownloadTime method” on page 113](#)
- ◆ [“CountUploadRows method” on page 106](#)
- ◆ [“Schema property” on page 100](#)
- ◆ [“SYNC_ALL_DB field” on page 313](#)
- ◆ [“SYNC_ALL_PUBS field” on page 313](#)
- ◆ [“GetPublicationSchema method” on page 191](#)
- ◆ [“ULDatabaseSchema class” on page 185](#)

ULPublicationSchema members

Public static fields (shared)

Member name	Description
SYNC_ALL_DB field	Publication mask corresponding to the entire database.
SYNC_ALL_PUBS field	Publication mask corresponding to all publications.

Public properties

Member name	Description
IsOpen property	Determines whether the publication schema is open or closed.
Mask property	Returns the publication mask of the publication.
Name property	Returns the name of this publication.

SYNC_ALL_DB field

Publication mask corresponding to the entire database.

Syntax

Visual Basic

Public Shared **SYNC_ALL_DB** As Integer

C#

```
public const int SYNC_ALL_DB;
```

SYNC_ALL_PUBS field

Publication mask corresponding to all publications.

Syntax

Visual Basic

Public Shared **SYNC_ALL_PUBS** As Integer

C#

```
public const int SYNC_ALL_PUBS;
```

IsOpen property

Determines whether the publication schema is open or closed.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen {get;}
```

Property value

True if the publication schema is open, false if the publication schema closed.

Mask property

Returns the publication mask of the publication.

Syntax

Visual Basic

Public Readonly Property **Mask** As Integer

C#

```
public int Mask {get;}
```

Property value

The publication mask of the publication.

Remarks

Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name, or refresh the cached masks and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - If a SQL error exception occurs. PUBLICATION_NOT_FOUND is issued if a schema upgrade has deleted or renamed this publication.

Name property

Returns the name of this publication.

Syntax**Visual Basic**

Public Readonly Property **Name** As String

C#

public string **Name** {get;}

Property value

A string specifying the name of the publication.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred. PUBLICATION_NOT_FOUND is issued if a schema upgrade has deleted or renamed the publication.

ULResultSet class

UL Ext.: Represents an editable result set in an UltraLite database.

Syntax

Visual Basic

Public Class **ULResultSet**
Inherits ULDataReader

C#

```
public class ULResultSet :  
    ULDataReader
```

Remarks

There is no constructor for this class. ResultSets are created using the `ULCommand.ExecuteResultSet()` method of the `ULCommand` class.

```
' Visual Basic  
Dim cmd As ULCommand = new ULCommand( _  
    "SELECT emp_id FROM employee", conn _  
)  
Dim resultSet As ULResultSet = cmd.ExecuteResultSet()  
  
// C#  
ULCommand cmd = new ULCommand(  
    "SELECT emp_id FROM employee", conn  
);  
ULResultSet resultSet = cmd.ExecuteResultSet();
```

A `ULResultSet` object represents an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use `ULCommand.ExecuteTable()` or a `ULDataAdapter`.

Inherits: `ULDataReader`

Implements: `System.Data.IDataReader`, `System.Data.IDataRecord`, `System.IDisposable`

See also

- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“ExecuteResultSet method” on page 74](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULDataAdapter class” on page 153](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

ULResultSet members

Public properties

Member name	Description
Depth property (inherited from ULDataReader)	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property (inherited from ULDataReader)	Returns the number of columns in the cursor.
IsBOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is before the first row.
IsClosed property (inherited from ULDataReader)	Checks whether the cursor is currently open.
IsEOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is after the last row.
Item property (inherited from ULDataReader)	Returns the value of the specified column in its native format. In C#, this property is the indexer for the ULDataReader class.
Item property (inherited from ULDataReader)	Returns the value of the specified named column in its native format. In C#, this property is the indexer for the ULDataReader class.
RecordsAffected property (inherited from ULDataReader)	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property (inherited from ULDataReader)	UL Ext.: Returns the number of rows in the cursor.
Schema property (inherited from ULDataReader)	UL Ext.: Holds the schema of this cursor.

Public methods

Member name	Description
AppendBytes method	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.
AppendChars method	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDbType.LongVarchar column.
Close method (inherited from ULDataReader)	Closes the cursor.
Delete method	Deletes the current row.
Dispose method (inherited from ULDataReader)	Releases the unmanaged resources used by the ULDataReader and optionally releases the managed resources.

Member name	Description
GetBoolean method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.Boolean</code> .
GetByte method (inherited from ULDataReader)	Returns the value for the specified column as an unsigned 8-bit value (<code>System.Byte</code>).
GetBytes method (inherited from ULDataReader)	Copies a subset of the value for the specified <code>ULDbType.LongBinary</code> column, beginning at the specified offset, to the specified offset of the destination <code>System.Byte</code> array.
GetBytes method (inherited from ULDataReader)	UL Ext.: Returns the value for the specified column as an array of <code>System.Bytes</code> . Only valid for columns of type <code>ULDbType.Binary</code> , <code>ULDbType.LongBinary</code> , or <code>ULDbType.UniqueIdentifier</code> .
GetChar method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetChars method (inherited from ULDataReader)	Copies a subset of the value for the specified <code>ULDbType.LongVarChar</code> column, beginning at the specified offset, to the specified offset of the destination <code>System.Char</code> array.
GetData method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetDataTypeName method (inherited from ULDataReader)	Returns the name of the specified column's provider data type.
GetDateTime method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.DateTime</code> with millisecond accuracy.
GetDecimal method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.Decimal</code> .
GetDouble method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.Double</code> .
GetFieldType method (inherited from ULDataReader)	Returns the <code>System.Type</code> most appropriate for the specified column.
GetFloat method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.Single</code> .
GetGuid method (inherited from ULDataReader)	Returns the value for the specified column as a UUID (<code>System.Guid</code>).
GetInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a <code>System.Int16</code> .
GetInt32 method (inherited from ULDataReader)	Returns the value for the specified column as an <code>Int32</code> .
GetInt64 method (inherited from ULDataReader)	Returns the value for the specified column as an <code>Int64</code> .

Member name	Description
GetName method (inherited from ULDataReader)	Returns the name of the specified column.
GetOrdinal method (inherited from ULDataReader)	Returns the column ID of the named column.
GetSchemaTable method (inherited from ULDataReader)	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader.
GetString method (inherited from ULDataReader)	Returns the value for the specified column as a System.String.
GetTimeSpan method (inherited from ULDataReader)	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt16.
GetUInt32 method (inherited from ULDataReader)	Returns the value for the specified column as a UInt32.
GetUInt64 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt64.
GetValue method (inherited from ULDataReader)	Returns the value of the specified column in its native format.
GetValues method (inherited from ULDataReader)	Returns all the column values for the current row.
IsDBNull method (inherited from ULDataReader)	Checks whether the value from the specified column is NULL.
MoveAfterLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the first row of the cursor.
MoveLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method (inherited from ULDataReader)	UL Ext.: Positions the cursor relative to the current row.

Member name	Description
NextResult method (inherited from ULDataReader)	Advances the ULDataReader to the next result when reading the results of batch SQL statements.
Read method (inherited from ULDataReader)	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
SetBoolean method	Sets the value for the specified column using a System.Boolean .
SetByte method	Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).
SetBytes method	Sets the value for the specified column using an array of System.Bytes .
SetDateTime method	Sets the value for the specified column using a System.DateTime .
SetDBNull method	Sets a column to NULL.
SetDecimal method	Sets the value for the specified column using a System.Decimal .
SetDouble method	Sets the value for the specified column using a System.Double .
SetFloat method	Sets the value for the specified column using a System.Single .
SetGuid method	Sets the value for the specified column using a System.Guid .
SetInt16 method	Sets the value for the specified column using an System.Int16 .
SetInt32 method	Sets the value for the specified column using an System.Int32 .
SetInt64 method	Sets the value for the specified column using an Int64 .
SetString method	Sets the value for the specified column using a System.String .
SetTimeSpan method	Sets the value for the specified column using a System.TimeSpan .
SetToDefault method	Sets the value for the specified column to its default value.
SetUInt16 method	Sets the value for the specified column using a System.UInt16 .
SetUInt32 method	Sets the value for the specified column using an System.UInt32 .
SetUInt64 method	Sets the value for the specified column using a System.UInt64 .
Update method	Updates the current row with the current column values (specified using the set methods).

Protected methods

Member name	Description
Finalize method	Releases unmanaged resources and performs other cleanup operations before the ULResultSet is reclaimed by garbage collection.

AppendBytes method

Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.

Syntax**Visual Basic**

```
Public Sub AppendBytes( _  
    ByVal columnID As Integer, _  
    ByVal val As Byte(), _  
    ByVal srcOffset As Integer, _  
    ByVal count As Integer _  
)
```

C#

```
public void AppendBytes(  
    int columnID,  
    byte[] val,  
    int srcOffset,  
    int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The value to append to the current new value for the column.
- ◆ **srcOffset** The start position in the source array.
- ◆ **count** The number of bytes to be copied.

Remarks

The bytes at position *srcOffset* (starting from 0) through *srcOffset+count-1* of the array *val* are appended to the value for the specified column.

When inserting, ULTable.InsertBegin initializes the new value to the column's default value. The data in the row is not actually changed until you execute an ULTable.Insert, and that change is not made permanent until it is committed.

When updating, the first append on a column clears the current value prior to appending the new value.

If any of the following are true, a `ULException` with code `ULSQLCode.SQLE_INVALID_PARAMETER` is thrown and the destination is not modified:

- ◆ *val* is null.
- ◆ *srcOffset* is negative.
- ◆ *count* is negative.
- ◆ *srcOffset+count* is greater than *val.Length*.

For other errors, a `ULException` with the appropriate error code is thrown.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Byte](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“InsertBegin method” on page 434](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“FieldCount property” on page 199](#)

AppendChars method

Appends the specified subset of the specified array of `System.Chars` to the new value for the specified `ULDbType.LongVarchar` column.

Syntax

Visual Basic

```
Public Sub AppendChars( _  
    ByVal columnID As Integer, _  
    ByVal val As Char(), _  
    ByVal srcOffset As Integer, _  
    ByVal count As Integer _  
)
```

C#

```
public void AppendChars(  
    int columnID,
```

```
char[] val,  
int srcOffset,  
int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The value to append to the current new value for the column.
- ◆ **srcOffset** The start position in the source array.
- ◆ **count** The number of bytes to be copied.

Remarks

The characters at position *srcOffset* (starting from 0) through *srcOffset+count-1* of the array *val* are appended to the value for the specified column. When inserting, ULTable.InsertBegin initializes the new value to the column's default value. The data in the row is not actually changed until you execute an ULTable.Insert, and that change is not made permanent until it is committed.

When updating, the first append on a column clears the current value prior to appending the new value.

If any of the following is true, a ULException with code ULSQLCode.SQLE_INVALID_PARAMETER is thrown and the destination is not modified:

- ◆ *val* is null.
- ◆ *srcOffset* is negative.
- ◆ *count* is negative.
- ◆ *srcOffset+count* is greater than *value.Length*.

For other errors, a ULException with the appropriate error code is thrown.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [System.Char](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“InsertBegin method” on page 434](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“ULException class” on page 242](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)

- ◆ [“FieldCount property” on page 199](#)

Delete method

Deletes the current row.

Syntax

Visual Basic

```
Public Sub Delete()
```

C#

```
public void Delete();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“StartSynchronizationDelete method” on page 117](#)
- ◆ [“StopSynchronizationDelete method” on page 117](#)

Finalize method

Releases unmanaged resources and performs other cleanup operations before the ULResultSet is reclaimed by garbage collection.

Syntax

Visual Basic

```
Overrides Protected Sub Finalize()
```

C#

```
protected override void Finalize();
```

SetBoolean method

Sets the value for the specified column using a System.Boolean.

Syntax

Visual Basic

```
Public Sub SetBoolean( _  
    ByVal columnID As Integer, _
```

```
ByVal val As Boolean _  
)
```

C#

```
public void SetBoolean(  
    int columnID,  
    bool val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Boolean](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetByte method

Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).

Syntax**Visual Basic**

```
Public Sub SetByte(_  
    ByVal columnID As Integer, _  
    ByVal val As Byte _  
)
```

C#

```
public void SetByte(
```

```
    int columnID,  
    byte val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Byte](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetBytes method

Sets the value for the specified column using an array of `System.Bytes`.

Syntax

Visual Basic

```
Public Sub SetBytes( _  
    ByVal columnID As Integer, _  
    ByVal val As Byte() _  
)
```

C#

```
public void SetBytes(  
    int columnID,  
    byte[] val  
);
```


Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

Only suitable for columns of type `ULDbType.Binary` or `ULDbType.LongBinary`, or for columns of type `ULDbType.UniqueIdentifier` when `val` is of length 16. The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Byte](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)
- ◆ [“ULDbType enumeration” on page 238](#)

SetDateTime method

Sets the value for the specified column using a `System.DateTime`.

Syntax

Visual Basic

```
Public Sub SetDateTime( _  
    ByVal columnID As Integer, _  
    ByVal val As Date _  
)
```

C#

```
public void SetDateTime(  
    int columnID,  
    DateTime val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The set value is accurate to the millisecond. The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [DateTime](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetDBNull method

Sets a column to NULL.

Syntax

Visual Basic

```
Public Sub SetDBNull( _  
    ByVal columnID As Integer _  
)
```

C#

```
public void SetDBNull(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Remarks

The data is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“IsColumnNullable method” on page 454](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetDecimal method

Sets the value for the specified column using a System.Decimal.

Syntax

Visual Basic

```
Public Sub SetDecimal( _  
    ByVal columnID As Integer, _  
    ByVal val As Decimal _  
)
```

C#

```
public void SetDecimal(  
    int columnID,  
    decimal val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)

- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Decimal](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetDouble method

Sets the value for the specified column using a System.Double.

Syntax

Visual Basic

```
Public Sub SetDouble( _  
    ByVal columnID As Integer, _  
    ByVal val As Double _  
)
```

C#

```
public void SetDouble(  
    int columnID,  
    double val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Double](#)
- ◆ [“Insert method” on page 434](#)

- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetFloat method

Sets the value for the specified column using a System.Single.

Syntax

Visual Basic

```
Public Sub SetFloat( _  
    ByVal columnID As Integer, _  
    ByVal val As Single _  
)
```

C#

```
public void SetFloat(  
    int columnID,  
    float val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Single](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetGuid method

Sets the value for the specified column using a System.Guid.

Syntax

Visual Basic

```
Public Sub SetGuid( _  
    ByVal columnID As Integer, _  
    ByVal val As Guid _  
)
```

C#

```
public void SetGuid(  
    int columnID,  
    Guid val  
);
```

Parameters

- ♦ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ♦ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed. Only valid for columns of type ULDbType.UniqueIdentifier or for columns of type ULDbType.Binary with length 16.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

See also

- ♦ [“ULResultSet class” on page 316](#)
- ♦ [“ULResultSet members” on page 317](#)
- ♦ [“GetNewUUID method” on page 113](#)
- ♦ [“GetOrdinal method” on page 222](#)
- ♦ [“Schema property” on page 204](#)
- ♦ [“GetFieldType method” on page 216](#)
- ♦ [“GetColumnSize method” on page 150](#)
- ♦ [Guid](#)
- ♦ [“Insert method” on page 434](#)
- ♦ [“Update method” on page 341](#)
- ♦ [“ULDbType enumeration” on page 238](#)
- ♦ [“ULDbType enumeration” on page 238](#)
- ♦ [“FieldCount property” on page 199](#)

SetInt16 method

Sets the value for the specified column using an System.Int16.

Syntax

Visual Basic

```
Public Sub SetInt16( _  
    ByVal columnID As Integer, _  
    ByVal val As Short _  
)
```

C#

```
public void SetInt16(  
    int columnID,  
    short val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Int16](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetInt32 method

Sets the value for the specified column using an System.Int32.

Syntax

Visual Basic

```
Public Sub SetInt32( _  
    ByVal columnID As Integer, _  
    ByVal val As Integer _  
)
```

C#

```
public void SetInt32(  
    int columnID,  
    int val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [Int32](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetInt64 method

Sets the value for the specified column using an Int64.

Syntax

Visual Basic

```
Public Sub SetInt64( _  
    ByVal columnID As Integer, _
```



```
    ByVal val As Long _  
)
```

C#

```
public void SetInt64(  
    int columnID,  
    long val  
);
```

Parameters

- ♦ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ♦ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ♦ [ULException class](#) - A SQL error occurred.

See also

- ♦ [“ULResultSet class” on page 316](#)
- ♦ [“ULResultSet members” on page 317](#)
- ♦ [“GetOrdinal method” on page 222](#)
- ♦ [“Schema property” on page 204](#)
- ♦ [“GetFieldType method” on page 216](#)
- ♦ [Int64](#)
- ♦ [“Insert method” on page 434](#)
- ♦ [“Update method” on page 341](#)
- ♦ [“FieldCount property” on page 199](#)

SetString method

Sets the value for the specified column using a System.String.

Syntax**Visual Basic**

```
Public Sub SetString( _  
    ByVal columnID As Integer, _  
    ByVal val As String _  
)
```

C#

```
public void SetString(
```

```
    int columnID,  
    string val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetTimeSpan method

Sets the value for the specified column using a System.TimeSpan.

Syntax

Visual Basic

```
Public Sub SetTimeSpan( _  
    ByVal columnID As Integer, _  
    ByVal val As TimeSpan _  
)
```

C#

```
public void SetTimeSpan(  
    int columnID,  
    TimeSpan val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

- ◆ **val** The new value for the column.

Remarks

The set value is accurate to the millisecond and is normalized to a nonnegative value between 0 and 24 hours. The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [TimeSpan](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetToDefault method

Sets the value for the specified column to its default value.

Syntax

Visual Basic

```
Public Sub SetToDefault( _  
    ByVal columnID As Integer _  
)
```

C#

```
public void SetToDefault(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0, `ULDataReader.FieldCount-1`]. The first column in the cursor has an ID value of zero.

Remarks

The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [“GetColumnDefaultValue method” on page 445](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetUInt16 method

Sets the value for the specified column using a System.UInt16.

Syntax

Visual Basic

```
Public Sub SetUInt16( _  
    ByVal columnID As Integer, _  
    ByVal val As UInt16 _  
)
```

C#

```
public void SetUInt16(  
    int columnID,  
    ushort val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)

- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt16](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetUInt32 method

Sets the value for the specified column using an System.UInt32.

Syntax

Visual Basic

```
Public Sub SetUInt32( _  
    ByVal columnID As Integer, _  
    ByVal val As UInt32 _  
)
```

C#

```
public void SetUInt32(  
    int columnID,  
    uint val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt32](#)
- ◆ [“Insert method” on page 434](#)

- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

SetUInt64 method

Sets the value for the specified column using a System.UInt64.

Syntax

Visual Basic

```
Public Sub SetUInt64( _  
    ByVal columnID As Integer, _  
    ByVal val As UInt64 _  
)
```

C#

```
public void SetUInt64(  
    int columnID,  
    ulong val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULResultSet class” on page 316](#)
- ◆ [“ULResultSet members” on page 317](#)
- ◆ [“GetOrdinal method” on page 222](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“GetFieldType method” on page 216](#)
- ◆ [UInt64](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Update method” on page 341](#)
- ◆ [“FieldCount property” on page 199](#)

Update method

Updates the current row with the current column values (specified using the set methods).

Syntax

Visual Basic

```
Public Sub Update()
```

C#

```
public void Update();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

ULResultSetSchema class

UL Ext.: Represents the schema of an UltraLite result set.

Syntax

Visual Basic

NotInheritable Public Class **ULResultSetSchema**
Inherits ULCursorSchema

C#

```
public sealed class ULResultSetSchema :  
    ULCursorSchema
```

Remarks

There is no constructor for this class. A ULResultSetSchema object is attached to a result set as its ULDataReader.Schema.

A result set schema is only valid while the data reader is open.

Inherits: ULCursorSchema

See also

- ◆ [“ULResultSetSchema members” on page 342](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULDataReader class” on page 195](#)
- ◆ [“ULResultSetSchema class” on page 342](#)
- ◆ [“Schema property” on page 204](#)
- ◆ [“ULCursorSchema class” on page 144](#)

ULResultSetSchema members

Public properties

Member name	Description
ColumnCount property (inherited from ULCursorSchema)	Returns the number of columns in the cursor.
IsOpen property (inherited from ULCursorSchema)	Checks whether the cursor schema is currently open.
Name property	Returns the name of the cursor.

Public methods

Member name	Description
GetColumnID method (inherited from ULCursorSchema)	Returns the column ID of the named column.
GetColumnName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnPrecision method (inherited from ULCursorSchema)	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnScale method (inherited from ULCursorSchema)	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSize method (inherited from ULCursorSchema)	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnSQLName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnULDbType method (inherited from ULCursorSchema)	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetSchemaTable method (inherited from ULCursorSchema)	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

Name property

Returns the name of the cursor.

Syntax

Visual Basic

Public Readonly Property **Name** As String

C#

```
public string Name {get;}
```

Property value

The SQL statement that generated the ULResultSetSchema.

ULRowUpdatedEventArgs class

Provides data for the `ULDataAdapter.RowUpdated` event.

Syntax

Visual Basic

NotInheritable Public Class **ULRowUpdatedEventArgs**
Inherits RowUpdatedEventArgs

C#

public sealed class **ULRowUpdatedEventArgs** :
RowUpdatedEventArgs

Remarks

Inherits: `System.Data.Common.RowUpdatedEventArgs`

See also

- ◆ [“ULRowUpdatedEventArgs members” on page 344](#)
- ◆ [“RowUpdated event” on page 176](#)
- ◆ [RowUpdatedEventArgs](#)

ULRowUpdatedEventArgs members

Public constructors

Member name	Description
ULRowUpdatedEventArgs constructor	Initializes a new instance of the <code>ULRowUpdatedEventArgs</code> class.

Public properties

Member name	Description
Command property	Returns the <code>ULCommand</code> executed when <code>DbDataAdapter.Update (System.Data.DataRow[], System.Data.Common.DataTableMapping)</code> is called.
Errors (inherited from <code>RowUpdatedEventArgs</code>)	Gets any errors generated by the .NET Framework data provider when the RowUpdatedEventArgs.Command was executed.
RecordsAffected property	Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement. For SELECT statements this value is -1.
Row (inherited from <code>RowUpdatedEventArgs</code>)	Gets the DataRow sent through an <code>DbDataAdapter.Update</code> .

Member name	Description
StatementType (inherited from RowUpdatedEventArgs)	Gets the type of SQL statement executed.
Status (inherited from RowUpdatedEventArgs)	Gets the UpdateStatus of the RowUpdatedEventArgs.Command property.
TableMapping (inherited from RowUpdatedEventArgs)	Gets the DataTableMapping sent through an DbDataAdapter.Update .

ULRowUpdatedEventArgs constructor

Initializes a new instance of the ULRowUpdatedEventArgs class.

Syntax

Visual Basic

```
Public Sub New( _
    ByVal row As DataRow, _
    ByVal command As IDbCommand, _
    ByVal statementType As StatementType, _
    ByVal tableMapping As DataTableMapping _
)
```

C#

```
public ULRowUpdatedEventArgs(
    DataRow row,
    IDbCommand command,
    StatementType statementType,
    DataTableMapping tableMapping
);
```

Parameters

- ◆ **row** The System.Data.DataRow sent through an DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping).
- ◆ **command** The System.Data.IDbCommand executed when DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping) is called.
- ◆ **statementType** One of the System.Data.StatementType values that specifies the type of query executed.
- ◆ **tableMapping** The System.Data.Common.DataTableMapping sent through an DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping).

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 344](#)
- ◆ [“ULRowUpdatedEventArgs members” on page 344](#)
- ◆ [DataRow](#)

- ◆ [IDbCommand](#)
- ◆ [StatementType](#)
- ◆ [DataTableMapping](#)
- ◆ [DbDataAdapter.Update](#)

Command property

Returns the ULCommand executed when DbDataAdapter.Update(System.Data.DataRow [],System.Data.Common.DataTableMapping) is called.

Syntax

Visual Basic

Public Readonly Property **Command** As ULCommand

C#

public ULCommand **Command** {get;}

Property value

The ULCommand object executed by the update.

Remarks

This is the strongly-typed version of System.Data.Common.RowUpdatedEventArgs.Command.

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 344](#)
- ◆ [“ULRowUpdatedEventArgs members” on page 344](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [RowUpdatedEventArgs.Command](#)

RecordsAffected property

Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement. For SELECT statements this value is -1.

Syntax

Visual Basic

Public Readonly Property **RecordsAffected** As Integer

C#

public int **RecordsAffected** {get;}

Property value

The number of rows changed, inserted, or deleted; 0 if no rows were affected or the statement failed; and -1 for SELECT statements.

ULRowUpdatedEventHandler delegate

Represents the method that will handle the `ULDataAdapter.RowUpdated` event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatedEventHandler( _  
    ByVal sender As Object, _  
    ByVal e As ULRowUpdatedEventArgs _  
)
```

C#

```
public delegate void ULRowUpdatedEventHandler(  
    object sender,  
    ULRowUpdatedEventArgs e  
);
```

Parameters

- ◆ **sender** The connection sending the event.
- ◆ **e** The `ULRowUpdatedEventArgs` object that contains the event data.

See also

- ◆ [“RowUpdated event” on page 176](#)
- ◆ [“ULRowUpdatedEventArgs class” on page 344](#)

ULRowUpdatingEventArgs class

Provides data for the `ULDataAdapter.RowUpdating` event.

Syntax

Visual Basic

NotInheritable Public Class **ULRowUpdatingEventArgs**
Inherits RowUpdatingEventArgs

C#

```
public sealed class ULRowUpdatingEventArgs :  
    RowUpdatingEventArgs
```

Remarks

Inherits: `System.Data.Common.RowUpdatingEventArgs`

See also

- ◆ [“ULRowUpdatingEventArgs members” on page 349](#)
- ◆ [“RowUpdating event” on page 177](#)
- ◆ [RowUpdatingEventArgs](#)

ULRowUpdatingEventArgs members

Public constructors

Member name	Description
ULRowUpdatingEventArgs constructor	Initializes a new instance of the <code>ULRowUpdatingEventArgs</code> class.

Public properties

Member name	Description
Command property	Specifies the <code>ULCommand</code> to execute when performing the <code>DbDataAdapter.Update(System.Data.DataRow [], System.Data.Common.DataTableMapping)</code> .
Errors (inherited from <code>RowUpdatingEventArgs</code>)	Gets any errors generated by the .NET Framework data provider when the RowUpdatedEventArgs.Command executes.
Row (inherited from <code>RowUpdatingEventArgs</code>)	Gets the DataRow to send through an <code>DbDataAdapter.Update</code> .
StatementType (inherited from <code>RowUpdatingEventArgs</code>)	Gets the type of SQL statement to execute.

Member name	Description
Status (inherited from RowUpdatingEventArgs)	Gets the UpdateStatus of the RowUpdatedEventArgs.Command property.
TableMapping (inherited from RowUpdatingEventArgs)	Gets the DataTableMapping to send through the DbDataAdapter.Update .

ULRowUpdatingEventArgs constructor

Initializes a new instance of the [ULRowUpdatingEventArgs](#) class.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal row As DataRow, _  
    ByVal command As IDbCommand, _  
    ByVal statementType As StatementType, _  
    ByVal tableMapping As DataTableMapping _  
)
```

C#

```
public ULRowUpdatingEventArgs(  
    DataRow row,  
    IDbCommand command,  
    StatementType statementType,  
    DataTableMapping tableMapping  
);
```

Parameters

- ♦ **row** The [System.Data.DataRow](#) to update.
- ♦ **command** The [System.Data.IDbCommand](#) to execute during the update.
- ♦ **statementType** One of the [System.Data.StatementType](#) values that specifies the type of query executed.
- ♦ **tableMapping** The [System.Data.Common.DataTableMapping](#) sent through an [DbDataAdapter.Update\(System.Data.DataRow\[\], System.Data.Common.DataTableMapping\)](#).

See also

- ♦ [“ULRowUpdatingEventArgs class” on page 349](#)
- ♦ [“ULRowUpdatingEventArgs members” on page 349](#)
- ♦ [DataRow](#)
- ♦ [IDbCommand](#)
- ♦ [StatementType](#)
- ♦ [DataTableMapping](#)
- ♦ [DbDataAdapter.Update](#)

Command property

Specifies the ULCommand to execute when performing the DbDataAdapter.Update(System.Data.DataRow [],System.Data.Common.DataTableMapping).

Syntax

Visual Basic

Public Property **Command** As ULCommand

C#

public ULCommand **Command** {get;set;}

Property value

The ULCommand object to execute when updating.

Remarks

This is the strongly-typed version of System.Data.Common.RowUpdatingEventArgs.Command.

See also

- ◆ [“ULRowUpdatingEventArgs class” on page 349](#)
- ◆ [“ULRowUpdatingEventArgs members” on page 349](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [RowUpdatingEventArgs.Command](#)

ULRowUpdatingEventHandler delegate

Represents the method that will handle the `ULDataAdapter.RowUpdating` event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatingEventHandler( _  
    ByVal sender As Object, _  
    ByVal e As ULRowUpdatingEventArgs _  
)
```

C#

```
public delegate void ULRowUpdatingEventHandler(  
    object sender,  
    ULRowUpdatingEventArgs e  
);
```

Parameters

- ◆ **sender** The connection sending the event.
- ◆ **e** The `ULRowUpdatingEventArgs` object that contains the event data.

See also

- ◆ [“RowUpdating event” on page 177](#)
- ◆ [“ULRowUpdatingEventArgs class” on page 349](#)

ULRuntimeType enumeration

UL Ext.: Enumerates the types of UltraLite.NET runtimes.

Syntax

Visual Basic

Public Enum **ULRuntimeType**
Inherits Short

C#

```
public enum ULRuntimeType :  
short
```

Member name

Member name	Description
STANDALONE_UL	Selects the standalone UltraLite.NET runtime. The standalone runtime accesses databases directly. Databases are accessed more quickly this way, but cannot be shared.
UL_ENGINE_CLIENT	Selects the UltraLite engine runtime. The UltraLite.NET engine client communicates with the UltraLite engine to access databases. This means that databases can be shared by different applications.

See also

- ◆ [“RuntimeType property” on page 179](#)

ULServerSyncListener interface

UL Ext.: The listener interface for receiving server synchronization messages.

Syntax

Visual Basic

Public Interface **ULServerSyncListener**

C#

public interface **ULServerSyncListener**

ULServerSyncListener members

Public methods

Member name	Description
ServerSyncInvoked method	Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

ServerSyncInvoked method

Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

Syntax

Visual Basic

```
Public Sub ServerSyncInvoked( _  
    ByVal messageName As String _  
)
```

C#

```
public void ServerSyncInvoked(  
    string messageName  
);
```

Parameters

- ♦ **messageName** The name of the message sent to the application.

Remarks

This method is invoked by a separate thread. To avoid multi-threading issues, it should post an event to the UI. If you are using multi-threading, it is recommended that you use a separate connection and use the lock keyword to access any objects shared with the rest of the application.

Example

The following code fragments demonstrate how to receive a server synchronization request and perform a synchronization in the UI thread.

```
' Visual Basic
Imports iAnywhere.Data.UltraLite

Public Class MainWindow
    Inherits System.Windows.Forms.Form
    Implements ULServerSyncListener
    Private conn As ULConnection

    Public Sub New(ByVal args() As String)

        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        ULConnection.DatabaseManager.SetServerSyncListener( _
            "myCompany.mymsg", "myCompany.myapp", Me _
        )
        'Create Connection
    End Sub

    Protected Overrides Sub OnClosing( _
        ByVal e As System.ComponentModel.CancelEventArgs _
    )
        ULConnection.DatabaseManager.SetServerSyncListener( _
            Nothing, Nothing, Nothing _
        )
        MyBase.OnClosing(e)
    End Sub

    Public Sub ServerSyncInvoked(ByVal messageName As String) _
        Implements ULServerSyncListener.ServerSyncInvoked
        Me.Invoke(New EventHandler(AddressOf Me.ServerSyncAction))
    End Sub

    Public Sub ServerSyncAction( _
        ByVal sender As Object, ByVal e As EventArgs _
    )
        ' Do Server sync
        conn.Synchronize()
    End Sub
End Class

// C#
using iAnywhere.Data.UltraLite;
public class Form1 : System.Windows.Forms.Form, ULServerSyncListener
{
    private System.Windows.Forms.MainMenu mainMenu1;
    private ULConnection conn;

    public Form1()
    {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();
    }
}
```

```
//
// TODO: Add any constructor code after
// InitializeComponent call
//
ULConnection.DatabaseManager.SetServerSyncListener(
    "myCompnay.mymsg", "myCompany.myapp", this
);
// Create connection
...
}

protected override void Dispose( bool disposing )
{
    base.Dispose( disposing );
}

protected override void OnClosing(
    System.ComponentModel.CancelEventArgs e
)
{
    ULConnection.DatabaseManager.SetServerSyncListener(
        null, null, null
    );
    base.OnClosing(e);
}

public void ServerSyncInvoked( string messageName )
{
    this.Invoke( new EventHandler( ServerSyncHandler ) );
}

internal void ServerSyncHandler(object sender, EventArgs e)
{
    conn.Synchronize();
}
}
```

ULSQLCode enumeration

UL Ext.: Enumerates the SQL codes that may be reported by UltraLite.NET.

Syntax

Visual Basic

Public Enum **ULSQLCode**

C#

public enum **ULSQLCode**

Member name

Member name	Description
SQL_E_AGGREGATES_NOT_ALLOWED	See “Invalid use of an aggregate function” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_ALIAS_NOT_UNIQUE	See “Alias '%1' is not unique” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_ALIAS_NOT_YET_DEFINED	See “Definition for alias '%1' must appear before its first reference” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_AMBIGUOUS_INDEX_NAME	See “Index name '%1' is ambiguous” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_ARGUMENT_CANNOT_BE_NULL	See “Argument %1 of procedure '%2' cannot be NULL” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_BAD_ENCRYPTION_KEY	See “Incorrect or missing encryption key” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_BAD_PARAM_INDEX	See “Input parameter index out of range” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_CANNOT_ACCESS_FILESYSTEM	See “Unable to access the filesystem on the device” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_CANNOT_CHANGE_ML_REMOTE_ID	See “Cannot change the MobiLink remote id when the status of the last upload is unknown” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_CANNOT_EXECUTE_STMT	See “Statement cannot be executed” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_CANNOT_MODIFY	See “Cannot modify column '%1' in table '%2'” [<i>SQL Anywhere 10 - Error Messages</i>].
SQL_E_CANNOT_REGISTER_LISTENER	See “The specified listener could not be registered” [<i>SQL Anywhere 10 - Error Messages</i>].

Member name	Description
SQLE_CLIENT_OUT_OF_MEMORY	See “Client out of memory” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_AMBIGUOUS	See “Column '%1' found in more than one table -- need a correlation name” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_CANNOT_BE_NULL	See “Column '%1' in table '%2' cannot be NULL” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_IN_INDEX	See “Cannot alter a column in an index” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_NOT_FOUND	See “Column '%1' not found” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_NOT_INDEXED	See “Column '%1' not part of any indexes in its containing table” [SQL Anywhere 10 - Error Messages].
SQLE_COLUMN_NOT_STREAMABLE	See “The operation failed because column '%1's type does not support streaming” [SQL Anywhere 10 - Error Messages].
SQLE_COMMUNICATIONS_ERROR	See “Communication error” [SQL Anywhere 10 - Error Messages].
SQLE_CONNECTION_ALREADY_EXISTS	See “This connection already exists” [SQL Anywhere 10 - Error Messages].
SQLE_CONNECTION_NOT_FOUND	See “Connection not found” [SQL Anywhere 10 - Error Messages].
SQLE_CONNECTION_RESTORED	See “UltraLite connection was restored” [SQL Anywhere 10 - Error Messages].
SQLE_CONSTRAINT_NOT_FOUND	See “Constraint '%1' not found” [SQL Anywhere 10 - Error Messages].
SQLE_CONVERSION_ERROR	See “Cannot convert %1 to a %2” [SQL Anywhere 10 - Error Messages].
SQLE_COULD_NOT_FIND_FUNCTION	See “Could not find '%1' in dynamic library '%2'” [SQL Anywhere 10 - Error Messages].
SQLE_COULD_NOT_LOAD_LIBRARY	See “Could not load dynamic library '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_CURSOR_ALREADY_OPEN	See “Cursor already open” [SQL Anywhere 10 - Error Messages].
SQLE_CURSOR_NOT_DECLARED	See “Cursor has not been declared” [SQL Anywhere 10 - Error Messages].
SQLE_CURSOR_NOT_OPEN	See “Cursor not open” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQLC_CURSOR_RESTORED	See “UltraLite cursor (or result set or table) was restored” [SQL Anywhere 10 - Error Messages].
SQLC_CURSOROP_NOT_ALLOWED	See “Illegal cursor operation attempt” [SQL Anywhere 10 - Error Messages].
SQLC_DATABASE_ERROR	See “Internal database error %1 -- transaction rolled back” [SQL Anywhere 10 - Error Messages].
SQLC_DATABASE_NAME_REQUIRED	See “Database name required to start server” [SQL Anywhere 10 - Error Messages].
SQLC_DATABASE_NOT_CREATED	See “Database creation failed: %1” [SQL Anywhere 10 - Error Messages].
SQLC_DATATYPE_NOT_ALLOWED	See “Expression has unsupported data type” [SQL Anywhere 10 - Error Messages].
SQLC_DBSPACE_FULL	See “A dbspace has reached its maximum file size” [SQL Anywhere 10 - Error Messages].
SQLC_DESCRIBE_NONSELECT	See “Can only describe a SELECT statement” [SQL Anywhere 10 - Error Messages].
SQLC_DEVICE_ERROR	See “I/O error %1 -- transaction rolled back” [SQL Anywhere 10 - Error Messages].
SQLC_DEVICE_IO_FAILED	See “File I/O failed for '%1'” [SQL Anywhere 10 - Error Messages].
SQLC_DIV_ZERO_ERROR	See “Division by zero” [SQL Anywhere 10 - Error Messages].
SQLC_DOWNLOAD_CONFLICT	See “Download failed because of conflicts with existing rows” [SQL Anywhere 10 - Error Messages].
SQLC_DOWNLOAD_RESTART_FAILED	See “Unable to retry download because upload is not finished” [SQL Anywhere 10 - Error Messages].
SQLC_DROP_DATABASE_FAILED	See “An attempt to delete database '%1' failed” [SQL Anywhere 10 - Error Messages].
SQLC_DUPLICATE_CURSOR_NAME	See “The cursor name '%1' already exists” [SQL Anywhere 10 - Error Messages].
SQLC_DUPLICATE_FOREIGN_KEY	See “Foreign key '%1' for table '%2' duplicates an existing foreign key” [SQL Anywhere 10 - Error Messages].
SQLC_DUPLICATE_OPTION	See “Option '%1' specified more than once” [SQL Anywhere 10 - Error Messages].
SQLC_DYNAMIC_MEMORY_EXHAUSTED	See “Dynamic memory exhausted” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQLE_ENCRYPTION_INITIALIZATION_FAILED	See “Could not initialize the encryption DLL: '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_ENCRYPTION_NOT_ENABLED	See “Encryption has not been enabled” [SQL Anywhere 10 - Error Messages].
SQLE_ENCRYPTION_NOT_ENABLED_WARNING	See “Encryption has not been enabled” [SQL Anywhere 10 - Error Messages].
SQLE_ENGINE_ALREADY_RUNNING	See “Database server already running” [SQL Anywhere 10 - Error Messages].
SQLE_ERROR	See “Run time SQL error -- %1” [SQL Anywhere 10 - Error Messages].
SQLE_ERROR_CALLING_FUNCTION	See “Could not allocate resources to call external function” [SQL Anywhere 10 - Error Messages].
SQLE_ERROR_IN_ASSIGNMENT	See “Error in assignment” [SQL Anywhere 10 - Error Messages].
SQLE_EXPRESSION_ERROR	See “Invalid expression near '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_FEATURE_NOT_ENABLED	See “The method you attempted to invoke was not enabled for your application” [SQL Anywhere 10 - Error Messages].
SQLE_FILE_BAD_DB	See “Unable to start specified database: '%1' is not a valid database file” [SQL Anywhere 10 - Error Messages].
SQLE_FILE_IN_USE	See “Specified database file already in use” [SQL Anywhere 10 - Error Messages].
SQLE_FILE_NOT_DB	See “Unable to start specified database: '%1' is not a database” [SQL Anywhere 10 - Error Messages].
SQLE_FILE_VOLUME_NOT_FOUND	See “Specified file system volume not found for database '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_FILE_WRONG_VERSION	See “Unable to start specified database: '%1' was created by a different version of the software” [SQL Anywhere 10 - Error Messages].
SQLE_FOREIGN_KEY_NAME_NOT_FOUND	See “Foreign key name '%1' not found” [SQL Anywhere 10 - Error Messages].
SQLE_IDENTIFIER_TOO_LONG	See “Identifier '%1' too long” [SQL Anywhere 10 - Error Messages].
SQLE_INCORRECT_VOLUME_ID	See “Incorrect volume ID for '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_INDEX_NAME_NOT_UNIQUE	See “Index name '%1' not unique” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQLC_INDEX_NOT_FOUND	See “Cannot find index named '%1'” [SQL Anywhere 10 - Error Messages].
SQLC_INDEX_NOT_UNIQUE	See “Index '%1' for table '%2' would not be unique” [SQL Anywhere 10 - Error Messages].
SQLC_INTERRUPTED	See “Statement interrupted by user” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_CONSTRAINT_REF	See “Invalid reference to or operation on constraint '%1'” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_DESCRIPTOR_INDEX	See “Invalid descriptor index” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_DESCRIPTOR_NAME	See “Invalid SQL descriptor name” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_DISTINCT_AGGREGATE	See “Grouped query contains more than one distinct aggregate function” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_FOREIGN_KEY	See “No primary key value for foreign key '%1' in table '%2'” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_FOREIGN_KEY_DEF	See “Column '%1' in foreign key has a different definition than primary key” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_GROUP_SELECT	See “Function or column reference to '%1' must also appear in a GROUP BY” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_INDEX_TYPE	See “Index type specification of '%1' is invalid” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_LOGON	See “Invalid user ID or password” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_OPTION	See “Invalid option '%1' -- no PUBLIC setting exists” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_OPTION_SETTING	See “Invalid setting for option '%1'” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_OPTION_VALUE	See “'%1' is an invalid value for '%2'” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_ORDER	See “Invalid ORDER BY specification” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_PARAMETER	See “Invalid parameter” [SQL Anywhere 10 - Error Messages].
SQLC_INVALID_PARSE_PARAMETER	See “Parse error: %1” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQL_E_INVALID_PUBLICATION_MASK	See “The specified publication mask is invalid” [SQL Anywhere 10 - Error Messages].
SQL_E_INVALID_SQL_IDENTIFIER	See “Invalid SQL identifier” [SQL Anywhere 10 - Error Messages].
SQL_E_INVALID_STATEMENT	See “Invalid statement” [SQL Anywhere 10 - Error Messages].
SQL_E_INVALID_UNION	See “Select lists in UNION, INTERSECT, or EXCEPT do not match in length” [SQL Anywhere 10 - Error Messages].
SQL_E_KEYLESS_ENCRYPTION	See “Unable to perform requested operation since this database uses keyless encryption” [SQL Anywhere 10 - Error Messages].
SQL_E_LOCKED	See “User '%1' has the row in '%2' locked” [SQL Anywhere 10 - Error Messages].
SQL_E_MAX_ROW_SIZE_EXCEEDED	See “Maximum row size for table '%1' would be exceeded” [SQL Anywhere 10 - Error Messages].
SQL_E_MEMORY_ERROR	See “Memory error -- transaction rolled back” [SQL Anywhere 10 - Error Messages].
SQL_E_METHOD_CANNOT_BE_CALLED	See “Method '%1' cannot be called at this time” [SQL Anywhere 10 - Error Messages].
SQL_E_MIRROR_FILE_MISMATCH	See “The mirror '%1' does not match database '%2'” [SQL Anywhere 10 - Error Messages].
SQL_E_NAME_NOT_UNIQUE	See “Item '%1' already exists” [SQL Anywhere 10 - Error Messages].
SQL_E_NO_COLUMN_NAME	See “Derived table '%1' has no name for column %2” [SQL Anywhere 10 - Error Messages].
SQL_E_NO_CURRENT_ROW	See “No current row of cursor” [SQL Anywhere 10 - Error Messages].
SQL_E_NO_INDICATOR	See “No indicator variable provided for NULL result” [SQL Anywhere 10 - Error Messages].
SQL_E_NO_MATCHING_SELECT_ITEM	See “The select list for the derived table '%1' has no expression to match '%2'” [SQL Anywhere 10 - Error Messages].
SQL_E_NO_PRIMARY_KEY	See “Table '%1' has no primary key” [SQL Anywhere 10 - Error Messages].
SQL_E_NOERROR	SQL_E_NOERROR(0) - This code indicates that there was no error or warning.
SQL_E_NON_UPDATEABLE_COLUMN	See “Cannot update an expression” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQL_NON_UPDATEABLE_CURSOR	See “FOR UPDATE has been incorrectly specified for a READ ONLY cursor” [SQL Anywhere 10 - Error Messages].
SQL_NOT_IMPLEMENTED	See “Feature '%1' not implemented” [SQL Anywhere 10 - Error Messages].
SQL_NOT_SUPPORTED_IN_ULTRALITE	See “Feature not available with UltraLite” [SQL Anywhere 10 - Error Messages].
SQL_NOTFOUND	See “Row not found” [SQL Anywhere 10 - Error Messages].
SQL_ONLY_ONE_TABLE	See “INSERT/DELETE on cursor can modify only one table” [SQL Anywhere 10 - Error Messages].
SQL_OVERFLOW_ERROR	See “Value %1 out of range for destination” [SQL Anywhere 10 - Error Messages].
SQL_PAGE_SIZE_INVALID	See “Invalid database page size” [SQL Anywhere 10 - Error Messages].
SQL_PARTIAL_DOWNLOAD_NOT_FOUND	See “No partial download was found” [SQL Anywhere 10 - Error Messages].
SQL_PERMISSION_DENIED	See “Permission denied: %1” [SQL Anywhere 10 - Error Messages].
SQL_PRIMARY_KEY_NOT_UNIQUE	See “Primary key for table '%1' is not unique” [SQL Anywhere 10 - Error Messages].
SQL_PRIMARY_KEY_TWICE	See “Table cannot have two primary keys” [SQL Anywhere 10 - Error Messages].
SQL_PRIMARY_KEY_VALUE_REF	See “Primary key for row in table '%1' is referenced by foreign key '%2' in table '%3'” [SQL Anywhere 10 - Error Messages].
SQL_PUBLICATION_NOT_FOUND	See “Publication '%1' not found” [SQL Anywhere 10 - Error Messages].
SQL_PUBLICATION_PREDICATE_IGNORED	See “Publication predicates were not evaluated” [SQL Anywhere 10 - Error Messages].
SQL_RESOURCE_GOVERNOR_EXCEEDED	See “Resource governor for '%1' exceeded” [SQL Anywhere 10 - Error Messages].
SQL_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY	See “Row was dropped from table %1 to maintain referential integrity” [SQL Anywhere 10 - Error Messages].
SQL_ROW_EXCEEDS_PAGE_SIZE	See “A row cannot be stored because it exceeds the database page size” [SQL Anywhere 10 - Error Messages].
SQL_SCHEMA_UPGRADE_NOT_ALLOWED	See “A schema upgrade is not currently allowed” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQLE_SERVER_SYNCHRONIZATION_ERROR	See “Synchronization failed due to an error on the server: %1” [SQL Anywhere 10 - Error Messages].
SQLE_START_STOP_DATABASE_DENIED	See “Request to start/stop database denied” [SQL Anywhere 10 - Error Messages].
SQLE_STATEMENT_ERROR	See “SQL statement error” [SQL Anywhere 10 - Error Messages].
SQLE_STRING_RIGHT_TRUNCATION	See “Right truncation of string data” [SQL Anywhere 10 - Error Messages].
SQLE_SUBQUERY_SELECT_LIST	See “Subquery allowed only one select list item” [SQL Anywhere 10 - Error Messages].
SQLE_SYNC_INFO_INVALID	See “Information for synchronization is incomplete or invalid, check %1” [SQL Anywhere 10 - Error Messages].
SQLE_SYNC_INFO_REQUIRED	See “Information for synchronization was not provided” [SQL Anywhere 10 - Error Messages].
SQLE_SYNC_NOT_REENTRANT	See “Synchronization process was unable to re-enter synchronization” [SQL Anywhere 10 - Error Messages].
SQLE_SYNC_STATUS_UNKNOWN	See “The status of the last synchronization upload is unknown” [SQL Anywhere 10 - Error Messages].
SQLE_SYNTAX_ERROR	See “Syntax error near '%1' %2” [SQL Anywhere 10 - Error Messages].
SQLE_TABLE_ALREADY_INCLUDED	See “Table '%1' is already included” [SQL Anywhere 10 - Error Messages].
SQLE_TABLE_IN_USE	See “Table in use” [SQL Anywhere 10 - Error Messages].
SQLE_TABLE_NOT_FOUND	See “Table '%1' not found” [SQL Anywhere 10 - Error Messages].
SQLE_TOO_MANY_BLOB_REFS	See “Too many references to a BLOB” [SQL Anywhere 10 - Error Messages].
SQLE_TOO_MANY_CONNECTIONS	See “Database server connection limit exceeded” [SQL Anywhere 10 - Error Messages].
SQLE_TOO_MANY_PUBLICATIONS	See “Too many publications specified in publication mask” [SQL Anywhere 10 - Error Messages].
SQLE_TOO_MANY_TEMP_TABLES	See “Too many temporary tables in connection” [SQL Anywhere 10 - Error Messages].
SQLE_TOO_MANY_USERS	See “Too many users in database” [SQL Anywhere 10 - Error Messages].
SQLE_ULTRALITE_DATABASE_NOT_FOUND	See “The database '%1' was not found” [SQL Anywhere 10 - Error Messages].

Member name	Description
SQLE_ULTRALITE_OBJ_CLOSED	See “Invalid operation on a closed object” [SQL Anywhere 10 - Error Messages].
SQLE_ULTRALITE_WRITE_ACCESS_DENIED	See “Write access was denied” [SQL Anywhere 10 - Error Messages].
SQLE_UNABLE_TO_CONNECT	See “Database cannot be started -- %1” [SQL Anywhere 10 - Error Messages].
SQLE_UNABLE_TO_START_DATABASE	See “Unable to start specified database: %1” [SQL Anywhere 10 - Error Messages].
SQLE_UNABLE_TO_START_DATABASE_VER_NEWER	See “Unable to start specified database: Server must be upgraded to start database %1” [SQL Anywhere 10 - Error Messages].
SQLE_UNCOMMITTED_TRANSACTIONS	See “You cannot synchronize or upgrade with uncommitted transactions” [SQL Anywhere 10 - Error Messages].
SQLE_UNKNOWN_FUNC	See “Unknown function '%1'” [SQL Anywhere 10 - Error Messages].
SQLE_UNKNOWN_OPTION	See “'%1' is an unknown option” [SQL Anywhere 10 - Error Messages].
SQLE_UNKNOWN_USERID	See “User ID '%1' does not exist” [SQL Anywhere 10 - Error Messages].
SQLE_UNRECOGNIZED_OPTION	See “The option '%1' is not recognized” [SQL Anywhere 10 - Error Messages].
SQLE_UPLOAD_FAILED_AT_SERVER	See “Synchronization server failed to commit the upload” [SQL Anywhere 10 - Error Messages].
SQLE_VALUE_IS_NULL	See “Cannot return NULL result as requested data type” [SQL Anywhere 10 - Error Messages].
SQLE_VARIABLE_INVALID	See “Invalid host variable” [SQL Anywhere 10 - Error Messages].
SQLE_WRONG_NUM_OF_INSERT_COLS	See “Wrong number of values for INSERT” [SQL Anywhere 10 - Error Messages].
SQLE_WRONG_PARAMETER_COUNT	See “Wrong number of parameters to function '%1'” [SQL Anywhere 10 - Error Messages].

ULStreamErrorCode enumeration

UL Ext.: Enumerates the error codes that may be reported by streams during synchronization.

Syntax

Visual Basic

Public Enum **ULStreamErrorCode**
Inherits Short

C#

```
public enum ULStreamErrorCode :  
short
```

Member name

Member name	Description
ACTSYNC_NO_PORT	ActiveSync synchronization can only be initiated by ActiveSync itself, either by placing the device in its cradle or by selecting "Synchronize" from the ActiveSync Manager. To initiate a synchronization from an application, use the TCP/IP socket synchronization stream.
ACTSYNC_NOT_INSTALLED	The ActiveSync provider has not been installed. Run dbasinst to install it (see documentation for details).
CONNECT_TIMEOUT	The connection attempt timed out. Either the server is not running on the indicated host and port or the timeout value needs to be increased to allow more time to connect.
COULD_NOT_OPEN_FILE_FOR_WRITE	The specified file could not be opened for write. Make sure that this is the correct file and that no other application is using it.
CREATE_RANDOM_OBJECT	The secure network layer could not create a random-number-generating object. Free up system resources, reconnect and retry the operation.
DEQUEUEING_CONNECTION	The MobiLink server encountered an error while attempting to get a queued connection (synchronization) request. Free up system resources. If the problem persists, restart the MobiLink server.
DUN_DIAL_FAILED	Automatic dialup failed to establish connection to the specified dial up network.
DUN_NOT_SUPPORTED	An attempt to dialup has failed due to insufficient system support. On PocketPC you must use cellcore.dll and on Windows you must use wininet.dll from IE 4.0 or above. Dialup is not supported on other platforms.
END_READ	Unable to finish a sequence of reads from the network. See also: READ

Member name	Description
END_WRITE	Unable to finish a sequence of writes to the network. See also: WRITE
GENERATE_RANDOM	The secure network layer requires a random number but was unable to generate one. Free up system resources, reconnect and retry the operation.
HTTP_AUTHENTICATION_FAILED	The supplied userid and password were rejected. Check that they were entered correctly. If so, contact your systems administrator to ensure you have proper access.
HTTP_AUTHENTICATION_REQUIRED	An HTTP server or gateway requested HTTP authentication. Please supply a userid and password using the HTTP synchronization parameters http_userid and http_password.
HTTP_BAD_STATUS_CODE	Examine the status line to determine the cause of the failure.
HTTP_BUFFER_SIZE_OUT_OF_RANGE	Fix the HTTP buffer size. A valid buffer size is positive and not overly large for the host platform.
HTTP_CHUNK_LEN_BAD_CHARACTER	Try using a fixed length HTTP body.
HTTP_CHUNK_LEN_ENCODED_MISSING	Try using a fixed length HTTP body.
HTTP_CLIENT_ID_NOT_SET	The client id was not passed into the HTTP client code. Contact technical support for a fix.
HTTP_CONTENT_TYPE_NOT_SPECIFIED	An unknown content type was specified. Refer to the documentation and change the content type to one of the supported types.
HTTP_CRLF_ENCODED_MISSING	The proxy you are using may not be compatible with MobiLink. Please check your configuration.
HTTP_CRLF_MISSING	The proxy you are using may not be compatible with MobiLink. Please check your configuration.
HTTP_EXPECTED_POST	The proxy you are using may not be compatible with MobiLink. Please check your configuration.
HTTP_EXTRA_DATA_END_READ	Extra data has been introduced into the HTTP body. This may have been added by a proxy agent. Try eliminating the proxy.
HTTP_HEADER_PARSE_ERROR	An error occurred while trying to parse an HTTP header. The header may be malformed.
HTTP_INTERNAL_HEADER_STATE	There was a problem decoding the HTTP header. This is an internal error that should never occur. Please contact technical support.
HTTP_INTERNAL_REQUEST_TYPE	There was a problem determining the HTTP request type. This is an internal error that should never occur. Please contact technical support.

Member name	Description
HTTP_INVALID_CHARAC- TER	An unexpected character was read in an HTTP header. The header may be malformed or the other side may not be sending HTTP at all.
HTTP_INVALID_SESSION_K EY	An unknown session key type was specified. Refer to the documentation and change the session key type to one of the supported types.
HTTP_MALFORMED_SESSI ON_COOKIE	The HTTP cookie used to manage the synchronization session is corrupt. Determine where the cookie is being corrupted. The most likely cause is a client error, or perhaps an HTTP intermediary misbehaving.
HTTP_NO_CONTD_CON- NECTION	The server timed out while waiting for the next HTTP request from the remote site. Determine why this request failed to reach the server or try a persistent connection.
HTTP_NO_PASSWORD	A userid was supplied for HTTP authentication but no password. Both are required for authentication.
HTTP_NO_USERID	A password was supplied for HTTP authentication but no userid. Both are required for authentication.
HTTP_PROXY_AUTHENTIC ATION_FAILED	The supplied userid and password were rejected by the proxy server. Check that they were entered correctly. If so, contact your systems administrator to ensure you have proper access.
HTTP_PROXY_AUTHENTIC ATION_REQUIRED	An HTTP proxy requested HTTP authentication. Please supply a userid and password using the HTTP synchronization parameters http_proxy_userid and http_proxy_password.
HTTP_SERVER_AUTH_FAIL ED	The Authentication-Info header sent from the server contained an incorrect value, causing authentication to fail. Make sure that you are connecting to a legitimate HTTP server.
HTTP_UNABLE_TO_PARSE_ COOKIE	Determine where the set cookie header is being corrupted.
HTTP_UNKNOWN_TRANSF ER_ENCODING	Determine how the unknown transfer encoding is getting generated.
HTTP_UNSUPPORTED_AUT H_ALGORITHM	The HTTP Digest authentication algorithm requested by the server is unsupported. Only "MD5" and "MD5-sess" are supported.
HTTP_VERSION	The requested HTTP version is unsupported. Consult the documentation and specify a supported HTTP version. At the time of publication the supported HTTP versions are 1.0 and 1.1.
INCONSISTENT_FIPS	Use of the -fips switch on the MobiLink server command line requires that all secure streams be FIPS-compliant. If a secure stream is not configured with the fips option, it will automatically be FIPS-compliant (for example, fips=y). Either remove the fips option from the secure stream, or enable it with fips=y.
INIT_RANDOM	The secure network layer could not initialize its random number generator. Free up system resources, reconnect and retry the operation.

Member name	Description
INTERNAL	An internal error has occurred in the network layer. Please contact technical support.
INTERNAL_API	An internal error has occurred in the network layer. Please contact technical support.
INTERNAL_PROTOCOL_NOT_LOADED	A synchronization protocol could not be loaded. If you are using UltraLite, make sure you have called the proper ULEnable method.
INTERRUPTED	The current operation was interrupted by the caller.
INVALID_COMPRESSION_TYPE	The specified compression type was not recognized.
INVALID_LOCAL_PATH	The local path for the downloaded file is invalid. Consult the documentation for details.
INVALID_NETWORK_LIBRARY	The given network interface DLL or shared object could not be loaded, possibly because it is invalid or corrupt.
INVALID_SYNC_PROTOCOL	The specified protocol is not a valid synchronization protocol.
LIBRARY_ENTRY_POINT_NOT_FOUND	The indicated library entry point could not be found.
LOAD_LIBRARY_FAILURE	The indicated library could not be found in the path. If you are trying to use TLS encryption for synchronization, make sure you have acquired the proper license.
LOAD_NETWORK_LIBRARY	The network interface library could not be found and/or loaded. Please check the following: 1) The sockets layer is properly installed. The correct network interface library (or DLL or shared object) must be present and accessible. 2) There are enough system resources available. Free up system resources if they are running low.
MEMORY_ALLOCATION	The network layer was unable to allocate enough bytes of storage. Free up system memory and retry the operation. The technique used to free up system memory depends on the operating system and how it is configured. The simplest technique is to reduce the number of active processes. Consult your operating system documentation for details.
MISSING_PARAMETER	The specified parameter was expected but not supplied.
NETWORK_LIBRARY_VERSION_MISMATCH	A network interface DLL or shared object could not be loaded because it is the wrong version.
NO_ECC_FIPS	There was a problem performing the given compression operation. Please contact technical support.

Member name	Description
NONE	This code indicates there was either no network error, or an unknown network error occurred.
NOT_IMPLEMENTED	An unimplemented internal feature was requested. Please contact technical support.
PARAMETER	Network parameters are of the form "name=value;[name2=value2 [...]]". This code indicates an invalid parameter value. Consult the documentation for the corresponding parameter name, and correct the parameter value.
PARAMETER_NOT_BOOLEAN	Network parameters are of the form "name=value;[name2=value2 [...]]". The parameter value is not a boolean value. Locate the offending parameter specification and change the value of the parameter to either 0 (for off or false) or 1 (for on or true).
PARAMETER_NOT_HEX	Network parameters are of the form "name=value;[name2=value2 [...]]". The parameter value is not a hexadecimal (base 16) value. Locate the offending parameter specification and change the value of the parameter to a hexadecimal value.
PARAMETER_NOT_UINT32	Network parameters are of the form "name=value;[name2=value2 [...]]". The parameter value is not an unsigned integer. Locate the offending parameter specification and change the value of the parameter to an unsigned integer.
PARAMETER_NOT_UINT32_RANGE	Network parameters are of the form "name=value;[name2=value2 [...]]". The parameter value is not an unsigned integer value or range. Locate the offending parameter specification and change the value of the parameter to an unsigned integer or an unsigned range. An unsigned range has the form: NNN-NNN.
PARSE	Network parameters are of the form "name=value;[name2=value2 [...]]". Optionally, the entire list of parameters may be enclosed in parentheses. The given string does not follow this convention. Inspect the string, fix any formatting problems, and retry the operation.
PROTOCOL_ERROR	An unexpected value or token was read.

Member name	Description
READ	Unable to read the given number of bytes from the network layer. Note that reads may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part of a basic operation in the upper layer. The cause of a read error is usually one of the following: 1) The network had a problem that caused the read to fail. Reconnect and retry the operation. 2) The connection timed out. Reconnect and retry the operation. 3) The other side of the connection cleanly terminated the connection. Consult the client and/or server logs for errors that indicate why the connection has been dropped. Consult the output-log errors and fix the cause, then retry the operation. 4) The process at the other side of the connection was aborted. Consult the client and/or server output logs for errors that indicate why the process was aborted. If the process was shut down by other than normal means, there may not be any errors in its output log. Reconnect and retry the operation. 5) The system is low on resources, and cannot perform the read. Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.
READ_TIMEOUT	Unable to read the given number of bytes from the network layer in the given time. Check that the network is functioning correctly, and that the sending application is still running.
SACI_ERROR_CLIENT	An error has been reported by the SACI implementation.
SACI_ERROR_SERVER	The SACI network library is reporting an error. Refer to the provider of the SACI network library to resolve the problem.
SECURE_ADD_CERTIFICATE	The secure network layer was unable to add a certificate to a certificate chain. Free up system resources and retry the operation.
SECURE_ADD_TRUSTED_CERTIFICATE	The secure network layer was unable to add a trusted certificate to a certificate chain. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.
SECURE_CERTIFICATE_COMMON_NAME	The given common name is not in the certificate chain. Check the following: 1) The common name was properly entered. 2) The correct certificate file was specified. 3) The common name is in the certificate chain. You can verify this with the readcert utility.

Member name	Description
SECURE_CERTIFICATE_COMPANY_NAME	The given organization name is not in the certificate chain. Check the following: 1) The organization name was properly entered. 2) The correct certificate file was specified. 3) The organization name is in the certificate chain. You can verify this with the readcert utility.
SECURE_CERTIFICATE_COMPANY_UNIT	The given organization unit is not in the certificate chain. Check the following: 1) The in company name was properly entered. 2) The correct certificate file was specified. 3) The company name is in the certificate chain. You can verify this with the readcert utility.
SECURE_CERTIFICATE_COUNT	The given file does not contain a certificate. Check the following: 1) The certificate file name was properly specified. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s).
SECURE_CERTIFICATE_EXPIRED	A certificate in the certificate chain has expired. Obtain a new certificate with a later expiry date and retry the operation.
SECURE_CERTIFICATE_EXPIRY_DATE	A certificate's expiry date could not be read. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.
SECURE_CERTIFICATE_FILE_NOT_FOUND	The certificate file could not be opened. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to read the file. This only applies to operating systems having user and/or file permissions.

Member name	Description
SECURE_CERTIFICATE_NOT_TRUSTED	The server's certificate was not signed by a trusted authority. Check the following: 1) The certificate file name was properly specified. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The client's list of trusted root certificates includes the server's root certificate.
SECURE_CERTIFICATE_ROOT	The root certificate in the chain is invalid. At the time of publication, this error was defined but not used.
SECURE_CREATE_CERTIFICATE	The secure network layer was unable to allocate storage for a certificate. Free up system resources and retry the operation.
SECURE_CREATE_PRIVATE_KEY_OBJECT	The secure network layer was unable to create a private key object, prior to loading the private key. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.
SECURE_DUPLICATE_CONTEXT	The secure network layer was unable to duplicate a security context. Free up system resources and retry the operation.
SECURE_EXPORT_CERTIFICATE	The secure network layer was unable to copy a certificate. Free up system resources and retry the operation.
SECURE_HANDSHAKE	The secure handshake failed. Check the following: 1) On the client, the correct host machine and port number were specified. 2) On the server, the correct port number was specified. 3) The correct certificate file was specified, both on the client and on the server.
SECURE_IMPORT_CERT_FROM_SYSTEM_STORE	Failed to import a certificate from the system certificate store.
SECURE_IMPORT_CERTIFICATE	The secure network layer was unable to import a certificate. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s).
SECURE_NO_CERTS_IN_SYSTEM_STORE	No certificates were found in the system's certificate store.

Member name	Description
SECURE_NO_SERVER_CERTIFICATE	No server certificate was provided. A server certificate is required for secure communications. The file provided must contain the full chain of certificates for the server as well as its private key.
SECURE_NO_SERVER_CERTIFICATE_PASSWORD	No server certificate password was provided. This password is required to decrypt the server's encrypted private key.
SECURE_NO_TRUSTED_ROOTS	No trusted root certificates were provided. At least one trusted root certificate is required for secure communications.
SECURE_OPEN_SYSTEM_CERT_STORE	An attempt to open a system certificate store failed.
SECURE_READ_CERTIFICATE	The certificate file could not be read. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.
SECURE_READ_PRIVATE_KEY	The private key could not be read from the certificate file. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.
SECURE_REDUNDANT_SERVER_CERTIFICATE_PASSWORD	A password was specified when the server's private key wasn't encrypted by any password.
SECURE_SET_IO	The secure network layer was unable to attach to the network layer. Free up system resources and retry the operation.
SECURE_SET_PRIVATE_KEY	The private key could not be used. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.

Member name	Description
SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND	The certificate file could not be found. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.
SECURE_TRUSTED_CERTIFICATE_READ	The secure network layer was unable to read the trusted certificate file. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.
SEED_RANDOM	The secure network layer could not seed its random number generator. Free up system resources, reconnect and retry the operation.
SERVER_ERROR	The server reported an error. Contact the MobiLink administrator to learn more.
SHUTTING_DOWN	The MobiLink server encountered an error in the network layer during shutdown. It is possible that some network operations pending at the time of shutdown were affected.
SOCKET_BIND	The network layer was unable to bind a socket to the given port. Check the following. 1) (Server only) Verify that the port isn't already in use. If the port is in use, either shut down the application listening on that port, or specify a different port. 2) (Server only) Verify that there are no firewall restrictions on the use of the port. 3) (Client only) If the client_port option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (for example, NNN-NNN). If a range was specified, consider making it a wider range, or a different range. 4) (Client only) If the client_port option was used, verify that there are no firewall restrictions on the use of the port.

Member name	Description
SOCKET_CLEANUP	The network layer was unable to clean up the socket layer. This error should only occur after all connections are finished, so no current connections should be affected.
SOCKET_CLOSE	The network layer was unable to close a socket. The network session may or may not have terminated prematurely, due to pending writes that were not flushed. Check the following: 1) The other side of the network connection had any errors. 2) The other side of the connection is running normally. 3) The machine is still connected to the network, and the network is responsive.
SOCKET_CONNECT	The network layer was unable to connect a socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 3) The correct host machine and port were specified. 4) The host server is running normally and listening on the correct port. 5) The host machine is listening for the proper socket type (TCP/IP vs. UDP). 6) If the client_port option was used, verify that there are no firewall restrictions on the use of the port. 7) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 8) There are enough system resources available. Free up system resources if they are running low.
SOCKET_CREATE_TCPIP	The network layer was unable to create a TCP/IP socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 6) There are enough system resources available. Free up system resources if they are running low.

Member name	Description
SOCKET_CREATE_UDP	The network layer was unable to create a UDP socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 3) If the client_port option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (for example, NNN-NNN). If a range was specified, consider making it a wider range, or a different range. 4) If the client_port option was used, verify that there are no firewall restrictions on the use of the port. 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 6) There are enough system resources available. Free up system resources if they are running low.
SOCKET_GET_HOST_BY_ADDR	The network layer was unable to get the name of a host using its IP address. At the time of publication, this error was defined but not used.
SOCKET_GET_NAME	The network layer was unable to determine a socket's local name. In a TCP/IP connection, each end of the connection has a socket exclusively attached to a port. A socket's local name includes this port number, which is assigned by the network at connection time. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.
SOCKET_GET_OPTION	The network layer was unable to get a socket option. This error may be the first indication that a connection has been lost. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.

Member name	Description
SOCKET_HOST_NAME_NOT_FOUND	The given host name could not be found. Check the following: 1) The host name was correctly specified. 2) The host is accessible. Many systems include a "ping" utility that can be used to verify access to a named host. 3) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try specifying the host's IP number (for example, NNN.NNN.NNN.NNN) instead of the host name. 4) The HOSTS file contains an entry that maps the host name to an IP number.
SOCKET_LISTEN	The server is unable to listen on a socket. The backlog refers to the maximum number of queued connection requests that may be pending at any given time. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) There are no firewall or other restrictions preventing a socket listener from running on the current machine. 3) The backlog setting is within the limit, if any, on the machine. 4) There are enough system resources available. Free up system resources if they are running low.
SOCKET_LIVENESS_OUT_OF_RANGE	An invalid liveness timeout value was specified. The liveness timeout value must be an integer between zero and 65535.
SOCKET_LOCALHOST_NAME_NOT_FOUND	The network layer was unable to determine the IP address of "localhost". Check the following: 1) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try explicitly specifying the localhost IP number (usually 127.0.0.1) instead. 2) The HOSTS file contains an entry that maps the "localhost" name to an IP number. 3) There are enough system resources available. Free up system resources if they are running low.
SOCKET_PORT_OUT_OF_RANGE	An invalid port number was specified. The port number must be an integer between zero and 65535.
SOCKET_SELECT	The network layer encountered an error attempting to wait for a socket to be ready for reading or writing. Check the following: 1) The machine is connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.

Member name	Description
SOCKET_SET_OPTION	The network layer was unable to set a socket option. This error may be the first indication that a connection has been lost. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.
SOCKET_SHUTDOWN	The network layer was unable to shut down a socket. Check the following: 1) The machine is connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.
SOCKET_STARTUP	The network layer was unable to initialize the socket layer. Check the following: 1) The sockets layer is properly installed. The correct network interface library must be present and accessible. 2) The machine is connected to the network, and the network is responsive. 3) There are enough system resources available. Free up system resources if they are running low.
UNEXPECTED_HTTP_REQUEST_TYPE	The given HTTP request type was unexpected at this time. The most likely cause is an HTTP client that is not a MobiLink client.
UNRECOGNIZED_TLS_TYPE	The TLS type is invalid. Consult the documentation for valid types.
VALUE_OUT_OF_RANGE	The specified value was not in the range of acceptable values for that parameter. Check the documentation for the parameter to learn the acceptable range of values.
WOULD_BLOCK	A requested operation would block where blocking is undesirable or unexpected.

Member name	Description
WRITE	Unable to write the given number of bytes to the network layer. Note that writes may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part of a basic operation in the upper layer. The cause of a write error is usually one of the following: 1) The network had a problem that caused the write to fail. Reconnect and retry the operation. 2) The connection timed out. Reconnect and retry the operation. 3) The other side of the connection cleanly terminated the connection. Consult the client and/or server logs for errors that indicate why the connection has been dropped. Consult the output-log errors and fix the cause, then retry the operation. 4) The process at the other side of the connection was aborted. Consult the client and/or server output logs for errors that indicate why the process was aborted. If the process was shut down by other than normal means, there may not be any errors in its output log. Reconnect and retry the operation. 5) The system is low on resources, and cannot perform the write. Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.
WRITE_TIMEOUT	Unable to write the given number of bytes to the network layer in the given time. Check that the network is functioning correctly, and that the receiving application is still running.

See also

- ◆ [“StreamErrorCode property” on page 415](#)

ULStreamErrorContext enumeration

UL Ext.: Enumerates the basic network operation being performed when the stream errors occurred.

Syntax

Visual Basic

Public Enum **ULStreamErrorContext**
Inherits Short

C#

```
public enum ULStreamErrorContext :  
short
```

Member name

Member name	Description
CLOSE	CLOSE(6)
CREATE	CREATE(3)
DESTROY	DESTROY(4)
END_READ	END_READ(11)
END_WRITE	END_WRITE(10)
GETVALUE	GETVALUE(14)
OPEN	OPEN(5)
READ	READ(7)
REGISTER	REGISTER(1)
SOFTSHUTDOWN	SOFTSHUTDOWN(13)
UNKNOWN	UNKNOWN(0)
UNREGISTER	UNREGISTER(2)
WRITE	WRITE(8)
WRITE_FLUSH	WRITE_FLUSH(9)
YIELD	YIELD(12)

See also

- ◆ [“StreamErrorContext property” on page 416](#)

ULStreamErrorID enumeration

UL Ext.: Enumerates the network layers that may report errors during synchronization.

Syntax

Visual Basic

Public Enum **ULStreamErrorID**
Inherits Short

C#

public enum **ULStreamErrorID** :
short

Member name

Member name	Description
ACTIVESYNC	ACTIVESYNC(22)
CERTICOM	CERTICOM(11)
CERTICOM_SSL	CERTICOM_SSL(13)
CERTICOM_TLS	CERTICOM_TLS(14)
DH_CAST	DH_CAST(9)
EMAIL	EMAIL(20)
FAKE	FAKE(2)
FILE	FILE(21)
HTTP	HTTP(7)
HTTPS	HTTPS(8)
JAVA_CERTICOM	JAVA_CERTICOM(12)
JAVA_RSA	JAVA_RSA(24)
NETTECH	NETTECH(5)
OPEN_SSL_RSA	OPEN_SSL_RSA(25)
PALM_CONDUIT	PALM_CONDUIT(3)
PALM_SS	PALM_SS(4)
PALM_SSL	PALM_SSL(26)
REPLAY	REPLAY(17)

Member name	Description
RIMBB	RIMBB(6)
RSA_TLS	RSA_TLS(23)
SECURE	SECURE(10)
SERIAL	SERIAL(1)
STRM	STRM(18)
TCPIP	TCPIP(0)
UDP	UDP(19)
WIRELESS	WIRELESS(16)
WIRESTRM	WIRESTRM(15)

See also

- ◆ [“StreamErrorID property” on page 416](#)

ULStreamType enumeration

UL Ext.: Enumerates the types of MobiLink synchronization streams to use for synchronization.

Syntax

Visual Basic

Public Enum **ULStreamType**
Inherits Short

C#

```
public enum ULStreamType :  
short
```

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams”](#) [*MobiLink - Client Administration*].

Separately licensed component required

ECC encryption and FIPS-certified encryption require a separate license. All strong encryption technologies are subject to export regulations.

See [“Separately licensed components”](#) [*SQL Anywhere 10 - Introduction*].

Member name

Member name	Description
HTTP	Synchronize via HTTP. The HTTP stream uses TCP/IP as its underlying transport. UltraLite applications act as Web browsers and the MobiLink server acts as a Web server. UltraLite applications send POST requests to send data to the server and GET requests to read data from the server.
HTTPS	Synchronize via HTTPS (HTTP with transport-layer security).
TCPIP	Synchronize via TCP/IP.
TLS	Synchronize via TCP/IP with transport layer security.

See also

- ♦ [“Stream property” on page 394](#)

ULSyncParms class

UL Ext.: Represents synchronization parameters that define how to synchronize an UltraLite database.

Syntax

Visual Basic

NotInheritable Public Class **ULSyncParms**

C#

public sealed class **ULSyncParms**

Remarks

There is no constructor for this class. Each connection has its own ULSyncParms instance, attached as its ULConnection.SyncParms.

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a ULSQLCode.SQLE_SYNC_INFO_INVALID SQLException is thrown by ULConnection.Synchronize().

Other sources of ULSQLCode.SQLE_SYNC_INFO_INVALID errors include not specifying a ULSyncParms.Stream value or a ULSyncParms.Version value.

See also

- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“ULConnection class” on page 90](#)
- ◆ [“SyncParms property” on page 101](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“DownloadOnly property” on page 388](#)
- ◆ [“PingOnly property” on page 391](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“Stream property” on page 394](#)
- ◆ [“Version property” on page 397](#)

ULSyncParms members

Public properties

Member name	Description
AuthenticationParms property	Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Member name	Description
CheckpointStore property	Specifies whether the client should perform extra store checkpoints to control the growth of the database store during synchronization.
DisableConcurrency property	Specifies whether concurrent access to UltraLite while performing a synchronization.
DownloadOnly property	Specifies whether to disable or enable uploads when synchronizing.
KeepPartialDownload property	Specifies whether to disable or enable partial downloads when synchronizing.
NewPassword property	Specifies a new MobiLink password for the user specified with UserName.
Password property	The MobiLink password for the user specified by UserName.
PingOnly property	Specifies whether the client should only ping the MobiLink server instead of performing a real synchronization.
PublicationMask property	Specifies the publications to be synchronized.
ResumePartialDownload property	Specifies whether to resume or discard a previous partial download.
SendColumnNames property	Specifies whether the client should send column names to the MobiLink server during synchronization.
SendDownloadAck property	Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization. The download acknowledgement is sent after the download has been fully applied and committed at the remote (a positive acknowledgement) or after the download fails (a negative acknowledgement).
Stream property	Specifies the MobiLink synchronization stream to use for synchronization.
StreamParms property	Specifies the parameters to configure the synchronization stream.
TableOrder property	Specifies the order tables should be uploaded to the consolidated database.
UploadOnly property	Specifies whether to disable or enable downloads when synchronizing.
UserName property	The user name that uniquely identifies the MobiLink client to the MobiLink server.
Version property	Specifies which synchronization script to use.

Public methods

Member name	Description
CopyFrom method	Copies the properties of the specified ULSyncParms object to this ULSyncParms object.

AuthenticationParms property

Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Syntax**Visual Basic**

Public Property **AuthenticationParms** As String()

C#

```
public string [] AuthenticationParms {get;set;}
```

Property value

An array of strings, each containing an authentication parameter (null array entries result in a synchronization error). The default is a null reference (Nothing in Visual Basic), meaning no authentication parameters.

Remarks

Only the first 255 strings are used and each string should be no longer than 128 characters (longer strings are truncated when sent to the MobiLink server).

CheckpointStore property

Specifies whether the client should perform extra store checkpoints to control the growth of the database store during synchronization.

Syntax**Visual Basic**

Public Property **CheckpointStore** As Boolean

C#

```
public bool CheckpointStore {get;set;}
```

Property value

True to specify that the client should perform extra store checkpoints. The default is false, meaning only required checkpointing is done.

Remarks

The checkpoint operation adds I/O operations for the application, and so slows synchronization. This option is most useful for large downloads with many updates. Devices with slow flash memory might not want to pay the performance penalty associated with additional checkpoints.

DisableConcurrency property

Specifies whether concurrent access to UltraLite while performing a synchronization.

Syntax**Visual Basic**

Public Property **DisableConcurrency** As Boolean

C#

```
public bool DisableConcurrency {get;set;}
```

Property value

True to disable concurrent access to UltraLite while performing a synchronization, false to enable concurrent access. The default is false.

Remarks

By default, other threads can perform UltraLite operations while a thread is synchronizing. When concurrent synchronization is disabled, other threads block on UltraLite calls until synchronization completes.

DownloadOnly property

Specifies whether to disable or enable uploads when synchronizing.

Syntax**Visual Basic**

Public Property **DownloadOnly** As Boolean

C#

```
public bool DownloadOnly {get;set;}
```

Property value

True to disable uploads when synchronizing, false to enable uploads. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“DownloadOnly property” on page 388](#)
- ◆ [“PingOnly property” on page 391](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“Synchronize method” on page 118](#)

KeepPartialDownload property

Specifies whether to disable or enable partial downloads when synchronizing.

Syntax**Visual Basic**

Public Property **KeepPartialDownload** As Boolean

C#

```
public bool KeepPartialDownload {get;set;}
```

Property value

True to enable partial downloads when synchronizing, false to disable partial downloads. The default is false.

Remarks

UltraLite.NET has the ability to restart downloads that fail because of communication errors or user aborts through the ULSyncProgressListener. UltraLite.NET processes the download as it is received. If a download is interrupted, then the partial download transaction remains in the database and can be resumed during the next synchronization.

To indicate that UltraLite.NET should save partial downloads, specify `connection.SyncParms.KeepPartialDownload=true`; otherwise the download is rolled back if an error occurs.

If a partial download was kept, then the output field `connection.SyncResult.ULSyncResult.PartialDownloadRetained` is set to true when `connection.Synchronize()` exits.

If `PartialDownloadRetained` is set, then you can resume a download. To do this, call `connection.Synchronize()` with `connection.SyncParms.ULSyncParms.ResumePartialDownload` set to true. It is recommended that you keep `KeepPartialDownload` set to true as well in case another communications error occurs. No upload is done if a download is skipped.

The download you receive during a resumed download is as old as when the download originally began. If you need the most up to date data, then you can do another download immediately after the special resumed download completes.

When resuming a download, many of the `ULSyncParms` fields are not relevant. For example, the `PublicationMask` field is not used. You receive the publications that you requested on the initial download. The only fields that need to be set are `ResumePartialDownload` and `UserName`. The fields `KeepPartialDownload` and `DisableConcurrency` can be set if desired and function as normal.

If you have a partial download and it is no longer needed, then you can call `ULConnection.RollbackPartialDownload()` to roll back the failed download transaction. Also, if you attempt to synchronize again and do not specify `ResumePartialDownload`, then the partial download is rolled back before the next synchronization begins.

For more information, see the [“Resuming failed downloads” \[MobiLink - Server Administration\]](#).

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“PartialDownloadRetained property” on page 415](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“RollbackPartialDownload method” on page 116](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UserName property” on page 397](#)
- ◆ [“DisableConcurrency property” on page 388](#)
- ◆ [“RollbackPartialDownload method” on page 116](#)

NewPassword property

Specifies a new MobiLink password for the user specified with `UserName`.

Syntax

Visual Basic

Public Property **NewPassword** As String

C#

```
public string NewPassword {get;set;}
```

Property value

A string specifying a new MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning the password is not changed.

Remarks

A new password takes effect after the next synchronization.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“UserName property” on page 397](#)

Password property

The MobiLink password for the user specified by UserName.

Syntax

Visual Basic

Public Property **Password** As String

C#

```
public string Password {get;set;}
```

Property value

A string specifying the MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning no password is specified.

Remarks

The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“NewPassword property” on page 390](#)
- ◆ [“UserName property” on page 397](#)

PingOnly property

Specifies whether the client should only ping the MobiLink server instead of performing a real synchronization.

Syntax

Visual Basic

Public Property **PingOnly** As Boolean

C#

```
public bool PingOnly {get;set;}
```

Property value

True to specify that the client should only ping the MobiLink server, false to specify the client should perform a real synchronization. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more

than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“DownloadOnly property” on page 388](#)
- ◆ [“PingOnly property” on page 391](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“Synchronize method” on page 118](#)

PublicationMask property

Specifies the publications to be synchronized.

Syntax**Visual Basic**

Public Property **PublicationMask** As Integer

C#

```
public int PublicationMask {get;set;}
```

Property value

A bitwise combination of publication masks, the special value `ULPublicationSchema.SYNC_ALL_PUBS`, or the special value `ULPublicationSchema.SYNC_ALL_DB`. The default is `ULPublicationSchema.SYNC_ALL_DB`. For more information on publication masks, see [ULPublicationSchema class](#).

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“Mask property” on page 314](#)
- ◆ [“SYNC_ALL_PUBS field” on page 313](#)
- ◆ [“SYNC_ALL_DB field” on page 313](#)
- ◆ [“SYNC_ALL_DB field” on page 313](#)
- ◆ [“ULPublicationSchema class” on page 312](#)

ResumePartialDownload property

Specifies whether to resume or discard a previous partial download.

Syntax

Visual Basic

Public Property **ResumePartialDownload** As Boolean

C#

```
public bool ResumePartialDownload {get;set;}
```

Property value

True to resume a previous partial download, false to discard a previous partial download. The default is false.

Remarks

Only at most one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

For more information on partial downloads, see [KeepPartialDownload property](#).

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“DownloadOnly property” on page 388](#)
- ◆ [“PingOnly property” on page 391](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“Synchronize method” on page 118](#)
- ◆ [“PartialDownloadRetained property” on page 415](#)

SendColumnNames property

Specifies whether the client should send column names to the MobiLink server during synchronization.

Syntax

Visual Basic

Public Property **SendColumnNames** As Boolean

C#

```
public bool SendColumnNames {get;set;}
```

Property value

True to specify that the client should send column names to the MobiLink server, false to specify that column names are not sent. The default is false.

Remarks

The column names are used by the MobiLink server for direct row handling. When the MobiLink server is using the row handling API to refer to columns by name rather than by index, you should set this option. This is the only use of the column names that are sent by this option.

SendDownloadAck property

Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization. The download acknowledgement is sent after the download has been fully applied and committed at the remote (a positive acknowledgement) or after the download fails (a negative acknowledgement).

Syntax**Visual Basic**

Public Property **SendDownloadAck** As Boolean

C#

```
public bool SendDownloadAck {get;set;}
```

Property value

Set True to specify that the client should send a download acknowledgement to the MobiLink server. Set False to specify that no download acknowledgement is sent. The default is False.

Remarks

If the client sends a download acknowledgement, the MobiLink server database worker thread must wait for the client to apply and commit the download. If the client does not sent a download acknowledgement, the MobiLink server is freed up sooner for its next synchronization.

Stream property

Specifies the MobiLink synchronization stream to use for synchronization.

Syntax**Visual Basic**

Public Property **Stream** As ULStreamType

C#

```
public ULStreamType Stream {get;set;}
```

Property value

One of the ULStreamType values specifying the type of synchronization stream to use. The default is ULStreamType.TCPIP.

Remarks

Most synchronization streams require parameters to identify the MobiLink server address and control other behavior. These parameters are supplied by ULSyncParms.StreamParms.

If the stream type is set to a value that is invalid for the platform, the stream type is set to ULSyncParms.TCPIP.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“ULStreamType enumeration” on page 384](#)
- ◆ [“StreamParms property” on page 395](#)
- ◆ [“ULStreamType enumeration” on page 384](#)

StreamParms property

Specifies the parameters to configure the synchronization stream.

Syntax

Visual Basic

Public Property **StreamParms** As String

C#

```
public string StreamParms {get;set;}
```

Property value

A string, in the form of a semicolon-separated list of keyword-value pairs, specifying the parameters for the stream. The default is a null reference (Nothing in Visual Basic).

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams” \[MobiLink - Client Administration\]](#).

StreamParms is a string containing all the parameters used for synchronization streams. Parameters are specified as a semicolon-separated list of name=value pairs ("param1=value1;param2=value2").

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“Stream property” on page 394](#)
- ◆ [“ULStreamType enumeration” on page 384](#)

TableOrder property

Specifies the order tables should be uploaded to the consolidated database.

Syntax

Visual Basic

Public Property **TableOrder** As String

C#

public string **TableOrder** {get;set;}

Property value

A string, in the form of a comma-separated list of table names. Tables names can be quoted using either single or double quotes. The default is a null reference (Nothing in Visual Basic), which does not override the default ordering of tables.

Remarks

If the foreign keys on your consolidated database match the foreign keys on your remote UltraLite database and there are no foreign key cycles, then you most likely don't need to use this feature. However, if you have tables that are part of foreign key cycles then list all tables that are part of a cycle in the TableOrder field. If you have tables with different foreign key relationships on the consolidated then also list these tables in the TableOrder field.

All tables that you don't specify will be appropriately sorted based of the foreign keys defined in the remote database.

UploadOnly property

Specifies whether to disable or enable downloads when synchronizing.

Syntax

Visual Basic

Public Property **UploadOnly** As Boolean

C#

public bool **UploadOnly** {get;set;}

Property value

True to disable downloads, false to enable downloads. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a ULSQLCode.SQLE_SYNC_INFO_INVALID SQLException is thrown by ULConnection.Synchronize().

See also

- ◆ [“ULSyncParms class” on page 385](#)

- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“DownloadOnly property” on page 388](#)
- ◆ [“PingOnly property” on page 391](#)
- ◆ [“ResumePartialDownload property” on page 392](#)
- ◆ [“UploadOnly property” on page 396](#)
- ◆ [“ULSQLCode enumeration” on page 357](#)
- ◆ [“Synchronize method” on page 118](#)

UserName property

The user name that uniquely identifies the MobiLink client to the MobiLink server.

Syntax

Visual Basic

Public Property **UserName** As String

C#

```
public string UserName {get;set;}
```

Property value

A string specifying the user name. This parameter has no default value, and must be explicitly set.

Remarks

The MobiLink server uses this value to determine the download content, to record the synchronization state, and to recover from interruptions during synchronization. This user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“Password property” on page 391](#)

Version property

Specifies which synchronization script to use.

Syntax

Visual Basic

Public Property **Version** As String

C#

```
public string Version {get;set;}
```

Property value

A string specifying the version of the synchronization script to use. This parameter has no default value, and must be explicitly set.

Remarks

Each synchronization script in the consolidated database is marked with a version string. For example, there can be two different download_cursor scripts, with each one identified by a different version string. The version string allows an UltraLite application to choose from a set of synchronization scripts.

CopyFrom method

Copies the properties of the specified ULSyncParms object to this ULSyncParms object.

Syntax**Visual Basic**

```
Public Sub CopyFrom( _  
    ByVal src As ULSyncParms _  
)
```

C#

```
public void CopyFrom(  
    ULSyncParms src  
);
```

Parameters

- ◆ **src** The object to copy from.

See also

- ◆ [“ULSyncParms class” on page 385](#)
- ◆ [“ULSyncParms members” on page 385](#)
- ◆ [“ULSyncParms class” on page 385](#)

ULSyncProgressData class

UL Ext.: Returns synchronization progress monitoring data.

Syntax

Visual Basic

Public Class **ULSyncProgressData**

C#

public class **ULSyncProgressData**

See also

- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressListener interface” on page 409](#)

ULSyncProgressData members

Public static fields (shared)

Member name	Description
FLAG_IS_BLOCKING field	A flag indicating that the synchronization is blocked awaiting a response from the MobiLink server.

Public properties

Member name	Description
ErrorMessage property	Returns the error message describing the error that occurred during synchronization.
Flags property	Returns the current synchronization flags indicating additional information relating to the current state.
ReceivedBytes property	Returns the number of bytes received so far. This information is updated for all states.
ReceivedDeletes property	Returns the number of deleted rows received so far.
ReceivedInserts property	Returns the number of inserted rows received so far.
ReceivedUpdates property	Returns the number of updated rows received so far.
SentBytes property	Returns the number of bytes sent so far. This information is updated for all states.
SentDeletes property	Returns the number of deleted rows sent so far.
SentInserts property	Returns the number of inserted rows sent so far.

Member name	Description
SentUpdates property	Returns the number of updated rows sent so far.
SQLCode property	Returns the SQL code for synchronization.
State property	Returns the current synchronization state.
SyncParms property	Returns a reference to the connection's SyncParms object.
SyncResult property	Returns a reference to the connection's SyncResult object. This object is only updated with ULSyncProgressState enumeration and UL-SyncProgressState enumeration events.
TableCount property	A count of the tables sent or received (TableCount of TableTotal) so far.
TableIndex property	Returns the index of the table currently being synchronized (tables are numbered 1 to DatabaseSchema.TableCount).
TableTotal property	Returns the number of tables being synchronized.

FLAG_IS_BLOCKING field

A flag indicating that the synchronization is blocked awaiting a response from the MobiLink server.

Syntax

Visual Basic

```
Public Shared FLAG_IS_BLOCKING As Integer
```

C#

```
public const int FLAG_IS_BLOCKING;
```

ErrorMessage property

Returns the error message describing the error that occurred during synchronization.

Syntax

Visual Basic

```
Public Readonly Property ErrorMessage As String
```

C#

```
public string ErrorMessage {get;}
```

Property value

A string describing the error that occurred during synchronization.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

Flags property

Returns the current synchronization flags indicating additional information relating to the current state.

Syntax**Visual Basic**

Public Readonly Property **Flags** As Integer

C#

```
public int Flags {get;}
```

Property value

An integer containing a combination of flags or'ed together.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“FLAG_IS_BLOCKING field” on page 400](#)

ReceivedBytes property

Returns the number of bytes received so far. This information is updated for all states.

Syntax**Visual Basic**

Public Readonly Property **ReceivedBytes** As Long

C#

```
public long ReceivedBytes {get;}
```

Property value

The number of bytes received so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

ReceivedDeletes property

Returns the number of deleted rows received so far.

Syntax

Visual Basic

Public Readonly Property **ReceivedDeletes** As Integer

C#

```
public int ReceivedDeletes {get;}
```

Property value

The number of deleted rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

ReceivedInserts property

Returns the number of inserted rows received so far.

Syntax

Visual Basic

Public Readonly Property **ReceivedInserts** As Integer

C#

```
public int ReceivedInserts {get;}
```

Property value

The number of inserted rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

ReceivedUpdates property

Returns the number of updated rows received so far.

Syntax**Visual Basic**

Public Readonly Property **ReceivedUpdates** As Integer

C#

public int **ReceivedUpdates** {get;}

Property value

The number of updated rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SentBytes property

Returns the number of bytes sent so far. This information is updated for all states.

Syntax**Visual Basic**

Public Readonly Property **SentBytes** As Long

C#

public long **SentBytes** {get;}

Property value

The number of bytes sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SentDeletes property

Returns the number of deleted rows sent so far.

Syntax

Visual Basic

Public Readonly Property **SentDeletes** As Integer

C#

public int **SentDeletes** {get;}

Property value

The number of deleted rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SentInserts property

Returns the number of inserted rows sent so far.

Syntax

Visual Basic

Public Readonly Property **SentInserts** As Integer

C#

public int **SentInserts** {get;}

Property value

The number of inserted rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SentUpdates property

Returns the number of updated rows sent so far.

Syntax**Visual Basic**

Public Readonly Property **SentUpdates** As Integer

C#

public int **SentUpdates** {get;}

Property value

The number of updated rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SQLCode property

Returns the SQL code for synchronization.

Syntax**Visual Basic**

Public Readonly Property **SQLCode** As ULSQLCode

C#

public ULSQLCode **SQLCode** {get;}

Property value

The [ULSQLCode enumeration](#) value for any synchronization error.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

State property

Returns the current synchronization state.

Syntax**Visual Basic**

Public Readonly Property **State** As ULSyncProgressState

C#

```
public ULSyncProgressState State {get;}
```

Property value

One of the [ULSyncProgressState enumeration](#) values specifying the current synchronization state.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

SyncParms property

Returns a reference to the connection's SyncParms object.

Syntax

Visual Basic

```
Public Readonly Property SyncParms As ULSyncParms
```

C#

```
public ULSyncParms SyncParms {get;}
```

Property value

A reference to the [SyncParms property](#) object.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“SyncParms property” on page 101](#)

SyncResult property

Returns a reference to the connection's SyncResult object. This object is only updated with [ULSyncProgressState enumeration](#) and [ULSyncProgressState enumeration](#) events.

Syntax

Visual Basic

```
Public Readonly Property SyncResult As ULSyncResult
```

C#

```
public ULSyncResult SyncResult {get;}
```


Property value

A reference to the [SyncResult](#) property object.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

TableCount property

A count of the tables sent or received (TableCount of TableTotal) so far.

Syntax**Visual Basic**

Public Readonly Property **TableCount** As Integer

C#

```
public int TableCount {get;}
```

Property value

The count of tables sent or received.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

TableIndex property

Returns the index of the table currently being synchronized (tables are numbered 1 to DatabaseSchema.TableCount).

Syntax**Visual Basic**

Public Readonly Property **TableIndex** As Integer

C#

```
public int TableIndex {get;}
```

Property value

The index of the table currently being synchronized. Tables are numbered 1 to [TableCount](#) property.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

TableTotal property

Returns the number of tables being synchronized.

Syntax

Visual Basic

Public Readonly Property **TableTotal** As Integer

C#

```
public int TableTotal {get;}
```

Property value

The number of tables being synchronized.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)
- ◆ [“ULSyncProgressData members” on page 399](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)
- ◆ [“ULSyncProgressState enumeration” on page 411](#)

ULSyncProgressListener interface

UL Ext.: The listener interface for receiving synchronization progress events.

Syntax

Visual Basic

Public Interface **ULSyncProgressListener**

C#

public interface **ULSyncProgressListener**

See also

- ◆ [“ULSyncProgressListener members” on page 409](#)
- ◆ [“Synchronize method” on page 119](#)

ULSyncProgressListener members

Public methods

Member name	Description
SyncProgressed method	Invoked during synchronization to inform the user of progress. This method should return true to cancel synchronization or return false to continue.

SyncProgressed method

Invoked during synchronization to inform the user of progress. This method should return true to cancel synchronization or return false to continue.

Syntax

Visual Basic

Public Function **SyncProgressed**(_
 ByVal *data* As ULSyncProgressData _
) As Boolean

C#

public bool **SyncProgressed**(
 ULSyncProgressData *data*
);

Parameters

- ◆ **data** A ULSyncProgressData object containing the latest synchronization progress data.

Return value

This method should return true to cancel synchronization or return false to continue.

Remarks

No UltraLite.NET API methods should be invoked during a SyncProgressed call.

See also

- ◆ [“ULSyncProgressListener interface” on page 409](#)
- ◆ [“ULSyncProgressListener members” on page 409](#)
- ◆ [“ULSyncProgressData class” on page 399](#)

ULSyncProgressState enumeration

UL Ext.: Enumerates all the states that can occur while synchronizing.

Syntax

Visual Basic

Public Enum **ULSyncProgressState**
Inherits Short

C#

public enum **ULSyncProgressState** :
short

Member name

Member name	Description
STATE_CANCELLED	Synchronization has been cancelled.
STATE_COMMITTING_DOWNLOAD	The download is being committed. The final count of rows received is included with this event. See ReceivedBytes property , ReceivedInserts property , ReceivedUpdates property , and ReceivedDeletes property .
STATE_CONNECTING	The synchronization stream has been built, but is not yet opened.
STATE_DISCONNECTING	The synchronization stream is about to be closed.
STATE_DONE	Synchronization has successfully completed. The connection's SyncResult property object has been updated.
STATE_ERROR	Synchronization has completed, but an error occurred. Check SyncResult property , ErrorMessage property , and SQLCode property for details.
STATE_FINISHING_UPLOAD	The upload is completing. The final count of rows sent is included with this event. See SentBytes property , SentInserts property , SentUpdates property , and SentDeletes property .
STATE_LAST	The last state entered by synchronization. This state is always entered and always after any other state. Other ULSyncProgressData fields may not contain valid data.
STATE_RECEIVING_DATA	Data for the current table is being received. ReceivedBytes property , ReceivedInserts property , ReceivedUpdates property , and ReceivedDeletes property have been updated.
STATE_RECEIVING_TABLE	A table is being received. Progress can be monitored using TableIndex property and TableCount property .
STATE_RECEIVING_UPLOAD_ACK	An acknowledgement that the upload is complete is being received.

Member name	Description
STATE_ROLLING_BACK_DOWNLOAD	Synchronization is rolling back the download because an error was encountered during the download. The error will be reported with a subsequent STATE_ERROR progress report.
STATE_SENDING_DATA	Data for the current table is being sent. SentBytes property , SentInserts property , SentUpdates property , and SentDeletes property have been updated.
STATE_SENDING_DOWNLOAD_ACK	An acknowledgement that the download is complete is being sent.
STATE_SENDING_HEADER	The synchronization stream has been opened and the header is about to be sent.
STATE_SENDING_TABLE	A table is being sent. Progress can be monitored using TableIndex property and TableCount property .
STATE_STARTING	No synchronization actions have been taken yet.

See also

- ◆ [“ULSyncProgressData class” on page 399](#)

ULSyncResult class

UL Ext.: Represents the status of the last synchronization.

Syntax

Visual Basic

Public Class **ULSyncResult**

C#

public class **ULSyncResult**

Remarks

There is no constructor for this class. Each connection has its own ULSyncResult instance, attached as its ULConnection.SyncResult. A ULSyncResult instance is only valid while that connection is open.

See also

- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“SyncResult property” on page 102](#)
- ◆ [“Synchronize method” on page 118](#)

ULSyncResult members

Public properties

Member name	Description
AuthStatus property	Returns the authorization status code for the last synchronization attempt.
AuthValue property	Returns the return value from custom user authentication synchronization scripts.
IgnoredRows property	Checks whether any uploaded rows were ignored during the last synchronization.
PartialDownloadRetained property	Checks whether a partial download was retained during the last synchronization.
StreamErrorCode property	Returns the error reported by the stream itself.
StreamErrorContext property	Returns the basic network operation being performed when the stream error occurred.
StreamErrorID property	Returns the ID of the network layer reporting an error.
StreamErrorSystem property	Returns the stream error system-specific code.
Timestamp property	Returns the timestamp of the last synchronization.

Member name	Description
UploadOK property	Checks whether the last upload synchronization was successful.

Protected methods

Member name	Description
Finalize method	Releases unmanaged resources and performs other cleanup operations before the ULSyncResult is reclaimed by garbage collection.

AuthStatus property

Returns the authorization status code for the last synchronization attempt.

Syntax**Visual Basic**

Public Readonly Property **AuthStatus** As ULAAuthStatusCode

C#

```
public ULAAuthStatusCode AuthStatus {get;}
```

Property value

One of the ULAAuthStatusCode values denoting the authorization status for the last synchronization attempt.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“ULAAuthStatusCode enumeration” on page 56](#)

AuthValue property

Returns the return value from custom user authentication synchronization scripts.

Syntax**Visual Basic**

Public Readonly Property **AuthValue** As Long

C#

```
public long AuthValue {get;}
```

Property value

A long integer returned from custom user authentication synchronization scripts.

IgnoredRows property

Checks whether any uploaded rows were ignored during the last synchronization.

Syntax

Visual Basic

Public Readonly Property **IgnoredRows** As Boolean

C#

```
public bool IgnoredRows {get;}
```

Property value

True if any uploaded rows were ignored during the last synchronization, false if no rows were ignored.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“DownloadOnly property” on page 388](#)

PartialDownloadRetained property

Checks whether a partial download was retained during the last synchronization.

Syntax

Visual Basic

Public Readonly Property **PartialDownloadRetained** As Boolean

C#

```
public bool PartialDownloadRetained {get;}
```

Property value

True if a download was interrupted and the partial download was retained, false if the download was not interrupted or if the partial download was rolled back.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“KeepPartialDownload property” on page 389](#)

StreamErrorCode property

Returns the error reported by the stream itself.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorCode** As UStreamErrorCode

C#

public UStreamErrorCode **StreamErrorCode** {get;}

Property value

One of the UStreamErrorCode values denoting the error reported by the stream itself, UStreamErrorCode.NONE if no error occurred.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“UStreamErrorCode enumeration” on page 366](#)
- ◆ [“UStreamErrorCode enumeration” on page 366](#)

StreamErrorContext property

Returns the basic network operation being performed when the stream error occurred.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorContext** As UStreamErrorContext

C#

public UStreamErrorContext **StreamErrorContext** {get;}

Property value

One of the UStreamErrorContext values denoting which basic network operation was being performed when the stream error occurred.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“UStreamErrorContext enumeration” on page 381](#)

StreamErrorID property

Returns the ID of the network layer reporting an error.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorID** As UStreamErrorID

C#

public UStreamErrorID **StreamErrorID** {get;}

Property value

One of the UStreamErrorID values denoting the ID of the network layer reporting an error.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [“UStreamErrorID enumeration” on page 382](#)

StreamErrorSystem property

Returns the stream error system-specific code.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorSystem** As Integer

C#

public int **StreamErrorSystem** {get;}

Property value

An integer denoting the stream error system-specific code.

Timestamp property

Returns the timestamp of the last synchronization.

Syntax**Visual Basic**

Public Readonly Property **Timestamp** As Date

C#

public DateTime **Timestamp** {get;}

Property value

A System.DateTime specifying the timestamp of the last synchronization.

See also

- ◆ [“ULSyncResult class” on page 413](#)
- ◆ [“ULSyncResult members” on page 413](#)
- ◆ [DateTime](#)

UploadOK property

Checks whether the last upload synchronization was successful.

Syntax

Visual Basic

Public Readonly Property **UploadOK** As Boolean

C#

```
public bool UploadOK {get;}
```

Property value

True if the last upload synchronization was successful, false if the last upload synchronization was unsuccessful.

Finalize method

Releases unmanaged resources and performs other cleanup operations before the ULSyncResult is reclaimed by garbage collection.

Syntax

Visual Basic

Overrides Protected Sub **Finalize()**

C#

```
protected override void Finalize();
```

ULTable class

UL Ext.: Represents a table in an UltraLite database.

Syntax

Visual Basic

Public Class **ULTable**
Inherits ULResultSet

C#

```
public class ULTable :  
    ULResultSet
```

Remarks

There is no constructor for this class. Tables are created using the `ULCommand.ExecuteTable()` of the `ULCommand`.

Inherits: `ULResultSet`

Implements: `System.Data.IDataReader`, `System.Data.IDataRecord`, `System.IDisposable`

See also

- ◆ [“ULTable members” on page 419](#)
- ◆ [“ExecuteTable method” on page 78](#)
- ◆ [“ULCommand class” on page 57](#)
- ◆ [“ULResultSet class” on page 316](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

ULTable members

Public properties

Member name	Description
Depth property (inherited from <code>ULDataReader</code>)	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property (inherited from <code>ULDataReader</code>)	Returns the number of columns in the cursor.
IsBOF property (inherited from <code>ULDataReader</code>)	UL Ext.: Checks whether the current row position is before the first row.
IsClosed property (inherited from <code>ULDataReader</code>)	Checks whether the cursor is currently open.

Member name	Description
IsEOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is after the last row.
Item property (inherited from ULDataReader)	Returns the value of the specified column in its native format. In C#, this property is the indexer for the ULDataReader class.
Item property (inherited from ULDataReader)	Returns the value of the specified named column in its native format. In C#, this property is the indexer for the ULDataReader class.
RecordsAffected property (inherited from ULDataReader)	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property (inherited from ULDataReader)	UL Ext.: Returns the number of rows in the cursor.
Schema property	Holds the table schema. This property is only valid while its connection is open.

Public methods

Member name	Description
AppendBytes method (inherited from ULResultSet)	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.
AppendChars method (inherited from ULResultSet)	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDbType.LongVarchar column.
Close method (inherited from ULDataReader)	Closes the cursor.
Delete method (inherited from ULResultSet)	Deletes the current row.
DeleteAllRows method	Deletes all rows in the table.
Dispose method (inherited from ULDataReader)	Releases the unmanaged resources used by the ULDataReader and optionally releases the managed resources.
FindBegin method	Prepares to perform a new Find on a table.
FindFirst method	Moves forward through the table from the beginning, looking for a row that exactly matches a value or full set of values in the current index.
FindFirst method	Moves forward through the table from the beginning, looking for a row that exactly matches a value or partial set of values in the current index.
FindLast method	Moves backward through the table from the end, looking for a row that exactly matches a value or full set of values in the current index.

Member name	Description
FindLast method	Moves backward through the table from the end, looking for a row that exactly matches a value or partial set of values in the current index.
FindNext method	Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or full set of values in the current index.
FindNext method	Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or partial set of values in the current index.
FindPrevious method	Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or full set of values in the current index.
FindPrevious method	Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or partial set of values in the current index.
GetBoolean method (inherited from ULDataReader)	Returns the value for the specified column as a System.Boolean.
GetByte method (inherited from ULDataReader)	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
GetBytes method (inherited from ULDataReader)	Copies a subset of the value for the specified ULDbType.LongBinary column, beginning at the specified offset, to the specified offset of the destination System.Byte array.
GetBytes method (inherited from ULDataReader)	UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.
GetChar method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetChars method (inherited from ULDataReader)	Copies a subset of the value for the specified ULDbType.LongVar- char column, beginning at the specified offset, to the specified offset of the destination System.Char array.
GetData method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetDataTypeName method (inherited from ULDataReader)	Returns the name of the specified column's provider data type.
GetDateTime method (inherited from ULDataReader)	Returns the value for the specified column as a System.DateTime with millisecond accuracy.
GetDecimal method (inherited from ULDataReader)	Returns the value for the specified column as a System.Decimal.

Member name	Description
GetDouble method (inherited from ULDataReader)	Returns the value for the specified column as a System.Double .
GetFieldType method (inherited from ULDataReader)	Returns the System.Type most appropriate for the specified column.
GetFloat method (inherited from ULDataReader)	Returns the value for the specified column as a System.Single .
GetGuid method (inherited from ULDataReader)	Returns the value for the specified column as a UUID (System.Guid).
GetInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.Int16 .
GetInt32 method (inherited from ULDataReader)	Returns the value for the specified column as an Int32 .
GetInt64 method (inherited from ULDataReader)	Returns the value for the specified column as an Int64 .
GetName method (inherited from ULDataReader)	Returns the name of the specified column.
GetOrdinal method (inherited from ULDataReader)	Returns the column ID of the named column.
GetSchemaTable method (inherited from ULDataReader)	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader .
GetString method (inherited from ULDataReader)	Returns the value for the specified column as a System.String .
GetTimeSpan method (inherited from ULDataReader)	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt16 .
GetUInt32 method (inherited from ULDataReader)	Returns the value for the specified column as a UInt32 .
GetUInt64 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt64 .
GetValue method (inherited from ULDataReader)	Returns the value of the specified column in its native format.
GetValues method (inherited from ULDataReader)	Returns all the column values for the current row.

Member name	Description
Insert method	Inserts a new row with the current column values (specified using the set methods). Each insert must be preceded by a call to ULTable.InsertBegin.
InsertBegin method	Prepares to insert a new row into the table by setting all current column values to their default values.
IsDBNull method (inherited from ULDataReader)	Checks whether the value from the specified column is NULL.
LookupBackward method	Moves backward through the table from the end, looking for a row that matches or is less than a value or full set of values in the current index.
LookupBackward method	Moves backward through the table from the beginning, looking for a row that matches or is less than a value or partial set of values in the current index.
LookupBegin method	Prepares to perform a new lookup on the table. The value(s) for which to search are specified by calling the appropriate setType method(s) on the columns in the index with which the table was opened.
LookupForward method	Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or full set of values in the current index.
LookupForward method	Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or partial set of values in the current index.
MoveAfterLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the first row of the cursor.
MoveLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method (inherited from ULDataReader)	UL Ext.: Positions the cursor relative to the current row.
NextResult method (inherited from ULDataReader)	Advances the ULDataReader to the next result when reading the results of batch SQL statements.

Member name	Description
Read method (inherited from ULDataReader)	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
SetBoolean method (inherited from ULResultSet)	Sets the value for the specified column using a System.Boolean .
SetByte method (inherited from ULResultSet)	Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).
SetBytes method (inherited from ULResultSet)	Sets the value for the specified column using an array of System.Bytes .
SetDateTime method (inherited from ULResultSet)	Sets the value for the specified column using a System.DateTime .
SetDBNull method (inherited from ULResultSet)	Sets a column to NULL.
SetDecimal method (inherited from ULResultSet)	Sets the value for the specified column using a System.Decimal .
SetDouble method (inherited from ULResultSet)	Sets the value for the specified column using a System.Double .
SetFloat method (inherited from ULResultSet)	Sets the value for the specified column using a System.Single .
SetGuid method (inherited from ULResultSet)	Sets the value for the specified column using a System.Guid .
SetInt16 method (inherited from ULResultSet)	Sets the value for the specified column using an System.Int16 .
SetInt32 method (inherited from ULResultSet)	Sets the value for the specified column using an System.Int32 .
SetInt64 method (inherited from ULResultSet)	Sets the value for the specified column using an Int64 .
SetString method (inherited from ULResultSet)	Sets the value for the specified column using a System.String .
SetTimeSpan method (inherited from ULResultSet)	Sets the value for the specified column using a System.TimeSpan .
SetToDefault method (inherited from ULResultSet)	Sets the value for the specified column to its default value.
SetUInt16 method (inherited from ULResultSet)	Sets the value for the specified column using a System.UInt16 .
SetUInt32 method (inherited from ULResultSet)	Sets the value for the specified column using an System.UInt32 .

Member name	Description
SetUInt64 method (inherited from ULResultSet)	Sets the value for the specified column using a System.UInt64.
Truncate method	Deletes all rows in the table while temporarily activating a stop synchronization delete.
Update method (inherited from ULResultSet)	Updates the current row with the current column values (specified using the set methods).
UpdateBegin method	Prepares to update the current row in the table.

Protected methods

Member name	Description
Finalize method	Releases unmanaged resources and performs other cleanup operations before the ULTable is reclaimed by garbage collection.

Schema property

Holds the table schema. This property is only valid while its connection is open.

Syntax

Visual Basic

Public Readonly Property **Schema** As ULTableSchema

C#

```
public ULTableSchema Schema {get;}
```

Property value

The ULTableSchema object representing the table schema.

Remarks

This property represents the complete schema of the table, including UltraLite.NET extended information which is not represented in the results from [ULDataReader.GetSchemaTable](#).

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“ULTableSchema class” on page 441](#)

DeleteAllRows method

Deletes all rows in the table.

Syntax**Visual Basic**

```
Public Sub DeleteAllRows()
```

C#

```
public void DeleteAllRows();
```

Remarks

In some applications, it can be useful to delete all rows from a table before downloading a new set of data into the table. Rows can be deleted from the UltraLite database without being deleted from the consolidated database using `ULConnection.StopSynchronizationDelete`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“Truncate method” on page 439](#)
- ◆ [“StopSynchronizationDelete method” on page 117](#)

Finalize method

Releases unmanaged resources and performs other cleanup operations before the `ULTable` is reclaimed by garbage collection.

Syntax**Visual Basic**

```
Overrides Protected Sub Finalize()
```

C#

```
protected override void Finalize();
```

FindBegin method

Prepares to perform a new Find on a table.

Syntax**Visual Basic**

```
Public Sub FindBegin()
```

C#

```
public void FindBegin();
```

Remarks

The value(s) for which to search are specified by calling the appropriate setType method(s) on the columns in the index with which the table was opened.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindFirst method” on page 427](#)
- ◆ [“FindFirst method” on page 428](#)
- ◆ [“FindLast method” on page 429](#)
- ◆ [“FindLast method” on page 430](#)

FindFirst method

Moves forward through the table from the beginning, looking for a row that exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **FindFirst()** As Boolean

C#

```
public bool FindFirst();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to FindBegin.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)

- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindBegin method” on page 426](#)
- ◆ [“FindNext method” on page 431](#)
- ◆ [“FindPrevious method” on page 432](#)
- ◆ [“FindFirst method” on page 428](#)
- ◆ [“IsEOF property” on page 200](#)
- ◆ [“FindBegin method” on page 426](#)

FindFirst method

Moves forward through the table from the beginning, looking for a row that exactly matches a value or partial set of values in the current index.

Syntax

Visual Basic

```
Overloads Public Function FindFirst( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindFirst(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to FindBegin.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindBegin method” on page 426](#)

- ◆ [“FindNext method” on page 431](#)
- ◆ [“FindPrevious method” on page 433](#)
- ◆ [“FindFirst method” on page 427](#)
- ◆ [“IsEOF property” on page 200](#)
- ◆ [“FindBegin method” on page 426](#)

FindLast method

Moves backward through the table from the end, looking for a row that exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **FindLast()** As Boolean

C#

```
public bool FindLast();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row found that exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to FindBegin.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindBegin method” on page 426](#)
- ◆ [“FindNext method” on page 431](#)
- ◆ [“FindPrevious method” on page 432](#)
- ◆ [“FindLast method” on page 430](#)
- ◆ [“IsBOF property” on page 199](#)
- ◆ [“FindBegin method” on page 426](#)

FindLast method

Moves backward through the table from the end, looking for a row that exactly matches a value or partial set of values in the current index.

Syntax

Visual Basic

```
Overloads Public Function FindLast( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindLast(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to find a value that matches based on the first column only, you should set the value for the first column, then supply a value of 1.

Return value

True if successful, false otherwise

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row found that exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to FindBegin.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindBegin method” on page 426](#)
- ◆ [“FindNext method” on page 431](#)
- ◆ [“FindPrevious method” on page 433](#)
- ◆ [“FindLast method” on page 429](#)
- ◆ [“IsBOF property” on page 199](#)
- ◆ [“FindBegin method” on page 426](#)

FindNext method

Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **FindNext()** As Boolean

C#

public bool **FindNext()**;

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the next row if it exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

FindNext behavior is undefined if the column values being searched for are modified during a row update.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindFirst method” on page 427](#)
- ◆ [“FindNext method” on page 431](#)
- ◆ [“IsEOF property” on page 200](#)

FindNext method

Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or partial set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **FindNext**(
 ByVal *numColumns* As Short
) As Boolean

C#

public bool **FindNext**(

```
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index, and you want to find a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the next row if it exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

FindNext behavior is undefined if the column values being searched for are modified during a row update.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindFirst method” on page 428](#)
- ◆ [“FindNext method” on page 431](#)
- ◆ [“FindFirst method” on page 427](#)
- ◆ [“IsEOF property” on page 200](#)

FindPrevious method

Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **FindPrevious()** As Boolean

C#

```
public bool FindPrevious();
```

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the previous row if it exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

FindPrevious behavior is undefined if the column values being searched for are modified during a row update.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindLast method” on page 429](#)
- ◆ [“FindPrevious method” on page 433](#)
- ◆ [“IsBOF property” on page 199](#)

FindPrevious method

Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or partial set of values in the current index.

Syntax

Visual Basic

```
Overloads Public Function FindPrevious( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindPrevious(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the previous row if it exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

FindPrevious behavior is undefined if the column values being searched for are modified during a row update.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“FindLast method” on page 429](#)
- ◆ [“FindLast method” on page 430](#)
- ◆ [“FindPrevious method” on page 432](#)
- ◆ [“IsBOF property” on page 199](#)

Insert method

Inserts a new row with the current column values (specified using the set methods).

Each insert must be preceded by a call to `ULTable.InsertBegin`.

Syntax

Visual Basic

```
Public Sub Insert()
```

C#

```
public void Insert();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“InsertBegin method” on page 434](#)

InsertBegin method

Prepares to insert a new row into the table by setting all current column values to their default values.

Syntax

Visual Basic

```
Public Sub InsertBegin()
```

C#

```
public void InsertBegin();
```

Remarks

Call the appropriate SetType or AppendType method(s) to specify the non-default values that are to be inserted.

The row is not actually inserted and the data in the row is not actually changed until you execute the Insert method, and that change is not made permanent until it is committed.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“Insert method” on page 434](#)
- ◆ [“Insert method” on page 434](#)

LookupBackward method

Moves backward through the table from the end, looking for a row that matches or is less than a value or full set of values in the current index.

Syntax**Visual Basic**

Overloads Public Function **LookupBackward()** As Boolean

C#

```
public bool LookupBackward();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is less than the index value. On failure, (no rows less than the value being looked for) the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to LookupBegin.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“LookupBegin method” on page 437](#)

- ◆ [“LookupBackward method” on page 436](#)
- ◆ [“IsBOF property” on page 199](#)

LookupBackward method

Moves backward through the table from the beginning, looking for a row that matches or is less than a value or partial set of values in the current index.

Syntax

Visual Basic

```
Overloads Public Function LookupBackward( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool LookupBackward(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the lookup. For example, if you have a three column index, and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is less than the index value. On failure, (no rows less than the value being looked for) the cursor position is before the first row (`ULDataReader.IsBOF`).

Each search must be preceded by a call to `LookupBegin`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“LookupBegin method” on page 437](#)
- ◆ [“LookupBackward method” on page 436](#)
- ◆ [“IsBOF property” on page 199](#)

LookupBegin method

Prepares to perform a new lookup on the table. The value(s) for which to search are specified by calling the appropriate setType method(s) on the columns in the index with which the table was opened.

Syntax

Visual Basic

```
Public Sub LookupBegin()
```

C#

```
public void LookupBegin();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“LookupForward method” on page 437](#)
- ◆ [“LookupForward method” on page 438](#)
- ◆ [“LookupBackward method” on page 435](#)
- ◆ [“LookupBackward method” on page 436](#)

LookupForward method

Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or full set of values in the current index.

Syntax

Visual Basic

```
Overloads Public Function LookupForward() As Boolean
```

C#

```
public bool LookupForward();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is greater than the index value. On failure, (no rows greater than the value being looked for) the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to ULTable.LookupBegin().

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“LookupBegin method” on page 437](#)
- ◆ [“LookupForward method” on page 438](#)
- ◆ [“IsEOF property” on page 200](#)

LookupForward method

Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or partial set of values in the current index.

Syntax

Visual Basic

Overloads Public Function **LookupForward**(_
 ByVal *numColumns* As Short _
) As Boolean

C#

```
public bool LookupForward(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the lookup. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is greater than the index value. On failure, (no rows greater than the value being looked for) the cursor position is after the last row `ULDataReader.IsEOF`.

Each search must be preceded by a call to `LookupBegin`.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)

- ◆ [“ULTable members” on page 419](#)
- ◆ [“LookupBegin method” on page 437](#)
- ◆ [“LookupForward method” on page 437](#)
- ◆ [“IsEOF property” on page 200](#)
- ◆ [“LookupBegin method” on page 437](#)

Truncate method

Deletes all rows in the table while temporarily activating a stop synchronization delete.

Syntax

Visual Basic

```
Public Sub Truncate()
```

C#

```
public void Truncate();
```

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“DeleteAllRows method” on page 425](#)

UpdateBegin method

Prepares to update the current row in the table.

Syntax

Visual Basic

```
Public Sub UpdateBegin()
```

C#

```
public void UpdateBegin();
```

Remarks

Column values are modified by calling the appropriate setType or AppendType method(s). The first append on a column clears the current column value prior to appending the new value.

The data in the row is not actually changed until you call ULResultSet.Update(), and that change is not made permanent until it is committed.

Modifying columns in the index used to open the table affects any active searches in unpredictable ways. Columns in the primary key of the table can not be updated.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTable class” on page 419](#)
- ◆ [“ULTable members” on page 419](#)
- ◆ [“Update method” on page 341](#)

ULTableSchema class

UL Ext.: Represents the schema of an UltraLite table.

Syntax

Visual Basic

```
NotInheritable Public Class ULTableSchema  
    Inherits ULCursorSchema
```

C#

```
public sealed class ULTableSchema :  
    ULCursorSchema
```

Remarks

There is no constructor for this class. A ULTableSchema object is attached to a table as its ULTable.Schema.

Inherits: ULCursorSchema

See also

- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“Schema property” on page 425](#)
- ◆ [“ULCursorSchema class” on page 144](#)

ULTableSchema members

Public properties

Member name	Description
ColumnCount property (inherited from ULCursorSchema)	Returns the number of columns in the cursor.
IndexCount property	Returns the number of indexes on the table.
IsNeverSynchronized property	Checks whether the table is marked as never being synchronized.
IsOpen property (inherited from ULCursorSchema)	Checks whether the cursor schema is currently open.
Name property	Returns the name of the table.
PrimaryKey property	Returns the index schema of the primary key for the table.
UploadUnchangedRows property	Checks whether the database uploads rows that have not changed.

Public methods

Member name	Description
GetColumnDefaultValue method	Returns the default value of the specified column.
GetColumnID method (inherited from ULCursorSchema)	Returns the column ID of the named column.
GetColumnName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnPartitionSize method	Returns the global autoincrement partition size assigned to the specified column.
GetColumnPrecision method (inherited from ULCursorSchema)	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnScale method (inherited from ULCursorSchema)	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSize method (inherited from ULCursorSchema)	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnSQLName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnULDbType method (inherited from ULCursorSchema)	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetIndex method	Returns the index schema of the named index.
GetIndexName method	Returns the name of the index identified by the specified index ID.
GetOptimalIndex method	The optimal index for searching a table using the specified column.
GetPublicationPredicate method	Returns the publication predicate for this table in the named publication.
GetSchemaTable method (inherited from ULCursorSchema)	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.
IsColumnAutoIncrement method	Checks whether the specified column's default is set to autoincrement.
IsColumnCurrentDate method	Checks whether the specified column's default is set to the current date (ULDbType.Date).
IsColumnCurrentTime method	Checks whether the specified column's default is set to the current time (ULDbType.Time).

Member name	Description
IsColumnCurrentTimestamp method	Checks whether the specified column's default is set to the current timestamp (ULDbType.TimeStamp).
IsColumnGlobalAutoIncrement method	Checks whether the specified column's default is set to global autoincrement.
IsColumnNewUUID method	Checks whether the specified column's default is set to a new UUID (System.Guid).
IsColumnNullable method	Checks whether the specified column is nullable.
IsInPublication method	Checks whether the table is contained in the named publication.

IndexCount property

Returns the number of indexes on the table.

Syntax

Visual Basic

Public Readonly Property **IndexCount** As Integer

C#

```
public int IndexCount {get;}
```

Property value

The number of indexes on the table or 0 if the table schema is closed.

Remarks

Index IDs range from 1 to IndexCount, inclusively.

Note: Index IDs and count may change during a schema upgrade. To correctly identify an index, access it by name or refresh the cached IDs and counts after a schema upgrade.

IsNeverSynchronized property

Checks whether the table is marked as never being synchronized.

Syntax

Visual Basic

Public Readonly Property **IsNeverSynchronized** As Boolean

C#

```
public bool IsNeverSynchronized {get;}
```

Property value

True if the table is marked as never being synchronized, false otherwise.

Remarks

Tables marked as never being synchronized are never synchronized, even if they are included in a publication. These tables are sometimes referred to as "no sync" tables.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

Name property

Returns the name of the table.

Syntax**Visual Basic**

Public Readonly Property **Name** As String

C#

public string **Name** {get;}

Property value

The name of the table as a string.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

PrimaryKey property

Returns the index schema of the primary key for the table.

Syntax**Visual Basic**

Public Readonly Property **PrimaryKey** As ULIndexSchema

C#

public ULIndexSchema **PrimaryKey** {get;}

Property value

A ULIndexSchema object representing the primary key for the table.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ULIndexSchema class” on page 265](#)

UploadUnchangedRows property

Checks whether the database uploads rows that have not changed.

Syntax**Visual Basic**

Public Readonly Property **UploadUnchangedRows** As Boolean

C#

```
public bool UploadUnchangedRows {get;}
```

Property value

True if the table is marked to always upload all rows during synchronization, false if the table is marked to upload only changed rows.

Remarks

Tables marked as such upload unchanged rows, as well as changed rows, when the table is synchronized. These tables are sometimes referred to as "all sync" tables.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

GetColumnDefaultValue method

Returns the default value of the specified column.

Syntax**Visual Basic**

```
Public Function GetColumnDefaultValue( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnDefaultValue(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in a table has an ID value of zero.

Return value

The default value of the specified column as a string or a null reference (Nothing in Visual Basic) if the default value is null.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)

GetColumnPartitionSize method

Returns the global autoincrement partition size assigned to the specified column.

Syntax**Visual Basic**

```
Public Function GetColumnPartitionSize( _  
    ByVal columnID As Integer _  
) As UInt64
```

C#

```
public ulong GetColumnPartitionSize(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

The column's global autoincrement partition size as a System.UInt64.

Remarks

All global autoincrement columns in a given table share the same global autoincrement partition.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“IsColumnGlobalAutoIncrement method” on page 452](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [UInt64](#)

GetIndex method

Returns the index schema of the named index.

Syntax**Visual Basic**

```
Public Function GetIndex( _  
    ByVal name As String _  
) As ULIndexSchema
```

C#

```
public ULIndexSchema GetIndex(  
    string name  
);
```

Parameters

- ◆ **name** The name of the index.

Return value

A ULIndexSchema object representing the named index.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ULIndexSchema class” on page 265](#)

GetIndexName method

Returns the name of the index identified by the specified index ID.

Syntax**Visual Basic**

```
Public Function GetIndexName( _
```

```
ByVal indexID As Integer _  
) As String
```

C#

```
public string GetIndexName(  
    int indexID  
);
```

Parameters

- ◆ **indexID** The ID of the index. The value must be in the range [1,IndexCount].

Return value

The name of the index as a string.

Remarks

Index IDs and counts may change during a schema upgrade. To correctly identify an index, access it by name or refresh the cached IDs and counts after a schema upgrade.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“IndexCount property” on page 443](#)

GetOptimalIndex method

The optimal index for searching a table using the specified column.

Syntax

Visual Basic

```
Public Function GetOptimalIndex( _  
    ByVal columnID As Integer _  
) As ULIndexSchema
```

C#

```
public ULIndexSchema GetOptimalIndex(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount>-1]. The first column in the table has an ID value of zero.

Return value

A ULIndexSchema object representing the optimal index for the specified column.

Remarks

The specified column is the first column in the index, but the index may have more than one column.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ULIndexSchema class” on page 265](#)

GetPublicationPredicate method

Returns the publication predicate for this table in the named publication.

Syntax**Visual Basic**

```
Public Function GetPublicationPredicate( _  
    ByVal pubName As String _  
) As String
```

C#

```
public string GetPublicationPredicate(  
    string pubName  
);
```

Parameters

- ◆ **pubName** The name of the publication.

Return value

The publication predicate as a string.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

IsColumnAutoIncrement method

Checks whether the specified column's default is set to autoincrement.

Syntax

Visual Basic

```
Public Function IsColumnAutoIncrement( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnAutoIncrement(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is autoincrementing, false if it is not autoincrementing.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)

IsColumnCurrentDate method

Checks whether the specified column's default is set to the current date (ULDbType.Date).

Syntax

Visual Basic

```
Public Function IsColumnCurrentDate( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentDate(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current date, false if the column does not default to the current date.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ULDbType enumeration” on page 238](#)

IsColumnCurrentTime method

Checks whether the specified column's default is set to the current time (ULDbType.Time).

Syntax**Visual Basic**

```
Public Function IsColumnCurrentTime( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentTime(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current time, false if the column does not default to the current time.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ULDbType enumeration” on page 238](#)

IsColumnCurrentTimestamp method

Checks whether the specified column's default is set to the current timestamp (ULDbType.TimeStamp).

Syntax

Visual Basic

```
Public Function IsColumnCurrentTimestamp( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentTimestamp(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current timestamp, false if the column does not default to the current timestamp.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [“ULDbType enumeration” on page 238](#)

IsColumnGlobalAutoIncrement method

Checks whether the specified column's default is set to global autoincrement.

Syntax

Visual Basic

```
Public Function IsColumnGlobalAutoIncrement( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnGlobalAutoIncrement(  
    int columnID  
);
```

```
int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is global autoincrementing, false if it is not global autoincrementing.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“GetColumnPartitionSize method” on page 446](#)
- ◆ [“DatabaseID property” on page 97](#)
- ◆ [“ColumnCount property” on page 145](#)

IsColumnNewUUID method

Checks whether the specified column's default is set to a new UUID (System.Guid).

Syntax

Visual Basic

```
Public Function IsColumnNewUUID( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnNewUUID(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to a new UUID, false if the column does not default to a new UUID.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)
- ◆ [Guid](#)

IsColumnNullable method

Checks whether the specified column is nullable.

Syntax**Visual Basic**

```
Public Function IsColumnNullable( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnNullable(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is nullable, false if it is not nullable.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

See also

- ◆ [“ULTableSchema class” on page 441](#)
- ◆ [“ULTableSchema members” on page 441](#)
- ◆ [“ColumnCount property” on page 145](#)

IsInPublication method

Checks whether the table is contained in the named publication.

Syntax**Visual Basic**

```
Public Function IsInPublication( _
```



```
    ByVal pubName As String _  
) As Boolean
```

C#

```
public bool IsInPublication(  
    string pubName  
);
```

Parameters

- ◆ **pubName** The name of the publication.

Return value

True if the table is in the publication, false if the table is not in the publication.

Exceptions

- ◆ [ULException class](#) - A SQL error occurred.

ULTransaction class

Represents a SQL transaction.

Syntax

Visual Basic

NotInheritable Public Class **ULTransaction**

C#

public sealed class **ULTransaction**

Remarks

There is no constructor for ULTransaction. To obtain a ULTransaction object, use the ULConnection.BeginTransaction(). To associate a command with a transaction, use the ULCommand.Transaction.

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

Implements: System.Data.IDbTransaction, System.IDisposable

See also

- ◆ [“ULTransaction members” on page 456](#)
- ◆ [“BeginTransaction method” on page 102](#)
- ◆ [“Transaction property” on page 68](#)
- ◆ [IDbTransaction](#)
- ◆ [IDisposable](#)

ULTransaction members

Public properties

Member name	Description
Connection property	Returns the connection associated with the transaction.
IsolationLevel property	Returns the isolation level for the transaction.

Public methods

Member name	Description
Commit method	Commits the database transaction.
Dispose method	Releases the resources used by the ULTransaction and rolls back any uncommitted commands.

Member name	Description
Rollback method	Rolls back the transaction's outstanding changes to the database.

Connection property

Returns the connection associated with the transaction.

Syntax

Visual Basic

Public Readonly Property **Connection** As ULConnection

C#

```
public ULConnection Connection {get;}
```

Property value

The ULConnection object associated with the transaction, or a null reference (Nothing in Visual Basic) if the transaction is no longer valid.

Remarks

This is the strongly-typed version of System.Data.IDbTransaction.Connection.

See also

- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“ULTransaction members” on page 456](#)
- ◆ [“BeginTransaction method” on page 102](#)
- ◆ [“ULConnection class” on page 90](#)
- ◆ [IDbTransaction.Connection](#)

IsolationLevel property

Returns the isolation level for the transaction.

Syntax

Visual Basic

NotOverridable Public Readonly Property **IsolationLevel** As IsolationLevel _
Implements IDbTransaction.IsolationLevel

C#

```
public IsolationLevel IsolationLevel {get;}
```

Property value

One of the System.Data.IsolationLevel values. UltraLite.NET only supports System.Data.IsolationLevel.ReadUncommitted.

Implements

[IDbTransaction.IsolationLevel](#)

See also

- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“ULTransaction members” on page 456](#)
- ◆ [“BeginTransaction method” on page 102](#)
- ◆ [IsolationLevel](#)
- ◆ [IsolationLevel.ReadUncommitted](#)

Commit method

Commits the database transaction.

Syntax

Visual Basic

```
NotOverridable Public Sub Commit() _  
    Implements IDbTransaction.Commit
```

C#

```
public void Commit();
```

Remarks

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

If Commit() fails due to a database error (for example, a referential integrity error), the transaction remains active. Correct the error and call the Commit() method again or call the ULTransaction.Rollback() to complete the transaction.

Implements

[IDbTransaction.Commit](#)

See also

- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“ULTransaction members” on page 456](#)
- ◆ [“Rollback method” on page 459](#)

Dispose method

Releases the resources used by the ULTransaction and rolls back any uncommitted commands.

Syntax

Visual Basic

```
NotOverridable Public Sub Dispose() _  
    Implements IDisposable.Dispose
```

C#

```
public void Dispose();
```

Implements

[IDisposable.Dispose](#)

Rollback method

Rolls back the transaction's outstanding changes to the database.

Syntax

Visual Basic

```
NotOverridable Public Sub Rollback() _  
    Implements IDbTransaction.Rollback
```

C#

```
public void Rollback();
```

Remarks

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

Implements

[IDbTransaction.Rollback](#)

See also

- ◆ [“ULTransaction class” on page 456](#)
- ◆ [“ULTransaction members” on page 456](#)
- ◆ [“Commit method” on page 458](#)

CHAPTER 6

UltraLite .NET 2.0 API Reference

Contents

ULActiveSyncListener interface	463
ULAuthStatusCode enumeration	467
ULBulkCopy class	468
ULBulkCopyColumnMapping class	480
ULBulkCopyColumnMappingCollection class	487
ULBulkCopyOptions enumeration	496
ULCommand class	497
ULCommandBuilder class	532
ULConnection class	542
ULConnectionParms class	578
ULConnectionParms.UnusedEventHandler delegate	589
ULConnectionStringBuilder class	590
ULCreateParms class	607
ULCursorSchema class	618
ULDataAdapter class	627
ULDatabaseManager class	639
ULDatabaseSchema class	646
ULDataReader class	656
ULDateOrder enumeration	694
ULDbType enumeration	695
ULException class	699
ULFactory class	703
ULFileTransfer class	709
ULFileTransferProgressData class	724
ULFileTransferProgressListener interface	727
ULIndexSchema class	729
ULInfoMessageEventArgs class	738
ULInfoMessageEventHandler delegate	741
ULMetaDataCollectionNames class	742

ULParameter class	751
ULParameterCollection class	768
ULPublicationSchema class	788
ULResultSet class	792
ULResultSetSchema class	816
ULRowsCopiedEventArgs class	819
ULRowsCopiedEventHandler delegate	822
ULRowUpdatedEventArgs class	823
ULRowUpdatedEventHandler delegate	827
ULRowUpdatingEventArgs class	828
ULRowUpdatingEventHandler delegate	831
ULRuntimeType enumeration	832
ULServerSyncListener interface	833
ULSQLCode enumeration	836
ULStreamErrorCode enumeration	846
ULStreamErrorContext enumeration	864
ULStreamErrorID enumeration	865
ULStreamType enumeration	867
ULSyncParms class	868
ULSyncProgressData class	883
ULSyncProgressListener interface	894
ULSyncProgressState enumeration	896
ULSyncResult class	898
ULTable class	904
ULTableSchema class	925
ULTransaction class	939

ULActiveSyncListener interface

UL Ext.: The listener interface for receiving ActiveSync events.

Syntax

Visual Basic

Public Interface **ULActiveSyncListener**

C#

public interface **ULActiveSyncListener**

See also

- ◆ [“ULActiveSyncListener members” on page 463](#)

ULActiveSyncListener members

Public methods

Member name	Description
ActiveSyncInvoked method	Invoked when the MobiLink provider for ActiveSync calls the application to perform synchronization.

See also

- ◆ [“ULActiveSyncListener interface” on page 463](#)

ActiveSyncInvoked method

Invoked when the MobiLink provider for ActiveSync calls the application to perform synchronization.

Syntax

Visual Basic

```
Public Sub ActiveSyncInvoked(  
    ByVal launchedByProvider As Boolean  
)
```

C#

```
public void ActiveSyncInvoked(  
    bool launchedByProvider  
);
```

Parameters

- ♦ **launchedByProvider** True if the application was launched by the MobiLink provider to perform ActiveSync synchronization. The application must then shut itself down after it has finished synchronizing. False if the application was already running when called by the MobiLink provider for ActiveSync.

Remarks

This method is invoked by a separate thread. To avoid multi-threading issues, it should post an event to the UI. If you are using multi-threading, it is recommended that you use a separate connection and use the lock keyword to access any objects shared with the rest of the application.

Once synchronization has completed, applications should call `ULDatabaseManager.SignalSyncIsComplete()` to signal the MobiLink provider for ActiveSync.

Example

The following code fragments demonstrate how to receive an ActiveSync request and perform a synchronization in the UI thread.

```
' Visual Basic
Imports iAnywhere.Data.UltraLite

Public Class MainWindow
    Inherits System.Windows.Forms.Form
    Implements ULAActiveSyncListener
    Private conn As ULConnection

    Public Sub New(ByVal args() As String)

        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        ULConnection.DatabaseManager.SetActiveSyncListener( _
            "myCompany.myapp", Me _
        )
        'Create Connection
        ...
    End Sub

    Protected Overrides Sub OnClosing( _
        ByVal e As System.ComponentModel.CancelEventArgs _
    )
        ULConnection.DatabaseManager.SetActiveSyncListener( _
            Nothing, Nothing _
        )
        MyBase.OnClosing(e)
    End Sub

    Public Sub ActiveSyncInvoked( _
        ByVal launchedByProvider As Boolean _
    ) Implements ULAActiveSyncListener.ActiveSyncInvoked
        Me.Invoke(New EventHandler(AddressOf Me.ActiveSyncAction))
    End Sub

    Public Sub ActiveSyncAction( _
```

```
        ByVal sender As Object, ByVal e As EventArgs _
    )
    ' Do active sync
    conn.Synchronize()
    ULConnection.DatabaseManager.SignalSyncIsComplete()
End Sub
End Class

// C#
using iAnywhere.Data.UltraLite;
public class Form1 : System.Windows.Forms.Form, ULActiveSyncListener
{
    private System.Windows.Forms.MainMenu mainMenu1;
    private ULConnection conn;

    public Form1()
    {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after
        // InitializeComponent call
        //
        ULConnection.DatabaseManager.SetActiveSyncListener(
            "myCompany.myapp", this
        );
        // Create connection
        ...
    }

    protected override void Dispose( bool disposing )
    {
        base.Dispose( disposing );
    }

    protected override void OnClosing(
        System.ComponentModel.CancelEventArgs e
    )
    {
        ULConnection.DatabaseManager.SetActiveSyncListener(
            null, null
        );
        base.OnClosing(e);
    }

    public void ActiveSyncInvoked(bool launchedByProvider)
    {
        this.Invoke( new EventHandler( ActiveSyncHandler ) );
    }

    internal void ActiveSyncHandler(object sender, EventArgs e)
    {
        conn.Synchronize();
        ULConnection.DatabaseManager.SignalSyncIsComplete();
    }
}
```

See also

- ◆ [“ULActiveSyncListener interface” on page 463](#)
- ◆ [“ULActiveSyncListener members” on page 463](#)
- ◆ [“SignalSyncIsComplete method” on page 645](#)

ULAuthStatusCode enumeration

UL Ext.: Enumerates the status codes that may be reported during MobiLink user authentication.

Syntax

Visual Basic

Public Enum **ULAuthStatusCode**

C#

public enum **ULAuthStatusCode**

Members

Member name	Description	Value
EXPIRED	User ID or password has expired - authorization failed (EXPIRED = 3).	3
IN_USE	User ID is already in use - authorization failed (IN_USE = 5).	5
INVALID	Bad user ID or password - authorization failed (INVALID = 4).	4
UNKNOWN	Authorization status is unknown, possibly because the connection has not yet performed a synchronization (UNKNOWN = 0).	0
VALID	User ID and password were valid at time of synchronization (VALID = 1).	1
VALID_BUT_EXPIRES_SOON	User ID and password were valid at time of synchronization, but will expire soon (VALID_BUT_EXPIRES_SOON = 2).	2

See also

- ◆ [“AuthStatus property” on page 899](#)

ULBulkCopy class

Efficiently bulk load an UltraLite table with data from another source. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULBulkCopy**
Implements IDisposable

C#

```
public sealed class ULBulkCopy : IDisposable
```

Remarks

Restrictions: The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopy members” on page 468](#)

ULBulkCopy members

Public constructors

Member name	Description
ULBulkCopy constructors	Initializes a ULBulkCopy object with the specified ULConnection.

Public properties

Member name	Description
BatchSize property	Gets or sets the number of rows in each batch. At the end of each batch, the rows in the batch are sent to the server.
BulkCopyTimeout property	Gets or sets the number of seconds for the operation to complete before it times out.
ColumnMappings property	Returns a collection of ULBulkCopyColumnMapping items. Column mappings define the relationships between columns in the data source and columns in the destination.
DestinationTableName property	Gets or sets the name of the destination table on the server.
NotifyAfter property	Specifies the number of rows to be processed before generating a notification event.

Public methods

Member name	Description
Close method	Closes the ULBulkCopy instance.
Dispose method	Disposes of the ULBulkCopy instance.
WriteToServer methods	Copies all rows in the supplied array of System.Data.DataRow objects to a destination table specified by the DestinationTableName field of the ULBulkCopy object.

Public events

Member name	Description
ULRowsCopied event	This event occurs every time the number of rows specified by NotifyAfter have been processed.

See also

- ◆ [“ULBulkCopy class” on page 468](#)

ULBulkCopy constructors

Initializes a ULBulkCopy object with the specified ULConnection.

See also

- ◆ [“ULConnection class” on page 542](#)

ULBulkCopy(ULConnection) constructor**Syntax****Visual Basic**

```
Public Sub New( _  
    ByVal connection As ULConnection _  
)
```

C#

```
public ULBulkCopy(  
    ULConnection connection  
);
```

Parameters

- ◆ **connection** The already open ULConnection that will be used to perform the bulk-copy operation. If the connection is not open, an exception is thrown in WriteToServer.

Remarks

Restrictions: The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“ULBulkCopy constructors” on page 469](#)
- ◆ [“ULConnection class” on page 542](#)

ULBulkCopy(String) constructor

Initializes a ULBulkCopy object with the specified connection string.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal connectionString As String _  
)
```

C#

```
public ULBulkCopy(  
    string connectionString  
);
```

Parameters

- ◆ **connectionString** The string defining the connection that will be opened for use by the ULBulkCopy instance. A connection string is a semicolon-separated list of keyword=value pairs.

For a list of parameters, see the [“ConnectionString property” on page 547](#).

Remarks

This syntax opens a connection during WriteToServer using connectionString. The connection is closed at the end of WriteToServer.

The connection string can be supplied using a ULConnectionParms object.

Restrictions: The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

Implements: System.IDisposable

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“ULBulkCopy constructors” on page 469](#)
- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [IDisposable](#)

ULBulkCopy(String, ULBulkCopyOptions) constructor

Initializes a ULBulkCopy object with the specified connection string and copy options.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal connectionString As String, _  
    ByVal copyOptions As ULBulkCopyOptions _  
)
```

C#

```
public ULBulkCopy(  
    string connectionString,  
    ULBulkCopyOptions copyOptions  
);
```

Parameters

- ◆ **connectionString** The string defining the connection that will be opened for use by the ULBulkCopy instance. A connection string is a semicolon-separated list of keyword=value pairs.

For a list of parameters, see the [“ConnectionString property” on page 547](#).

- ◆ **copyOptions** A combination of values from the ULBulkCopyOptions enumeration that determines how data source rows are copied to the destination table.

Remarks

This syntax opens a connection during WriteToServer using connectionString. The connection is closed at the end of WriteToServer.

Restrictions: The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“ULBulkCopy constructors” on page 469](#)
- ◆ [“ULBulkCopyOptions enumeration” on page 496](#)

ULBulkCopy(ULConnection, ULBulkCopyOptions, ULTransaction) constructor

Initializes a ULBulkCopy object with the specified ULConnection, copy options and ULTransaction.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal connection As ULConnection, _  
    ByVal copyOptions As ULBulkCopyOptions, _
```

```
ByVal externalTransaction As ULTransaction _  
)
```

C#

```
public ULBulkCopy(  
    ULConnection connection,  
    ULBulkCopyOptions copyOptions,  
    ULTransaction externalTransaction  
);
```

Parameters

- ◆ **connection** The already open ULConnection that will be used to perform the bulk-copy operation. If the connection is not open, an exception is thrown in WriteToServer.
- ◆ **copyOptions** A combination of values from the ULBulkCopyOptions enumeration that determines how data source rows are copied to the destination table.
- ◆ **externalTransaction** An existing ULTransaction instance under which the bulk copy will occur. If externalTransaction is not a null reference (Nothing in Visual Basic), then the bulk-copy operation is done within it. It is an error to specify both an external transaction and the ULBulkCopyOptions.UseInternalTransaction option.

Remarks

Restrictions: The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“ULBulkCopy constructors” on page 469](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULTransaction class” on page 939](#)

BatchSize property

Gets or sets the number of rows in each batch. At the end of each batch, the rows in the batch are sent to the server.

Syntax

Visual Basic

```
Public Property BatchSize As Integer
```

C#

```
public int BatchSize { get; set; }
```

Property value

The number of rows in each batch. The default is 0.

Remarks

Setting it to zero causes all the rows to be sent in one batch.

Setting it less than zero is an error.

If this value is changed while a batch is in progress, the current batch completes and any further batches use the new value.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)

BulkCopyTimeout property

Gets or sets the number of seconds for the operation to complete before it times out.

Syntax**Visual Basic**

Public Property **BulkCopyTimeout** As Integer

C#

```
public int BulkCopyTimeout { get; set; }
```

Property value

The default value is 30 seconds.

Remarks

A value of zero indicates no limit. This should be avoided because it may cause an indefinite wait.

If the operation times out, then all rows in the current transaction are rolled back and an `SAException` is raised.

Setting it less than zero is an error.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)

ColumnMappings property

Returns a collection of `ULBulkCopyColumnMapping` items. Column mappings define the relationships between columns in the data source and columns in the destination.

Syntax**Visual Basic**

Public Readonly Property **ColumnMappings** As ULBulkCopyColumnMappingCollection

C#

public ULBulkCopyColumnMappingCollection **ColumnMappings** { get; }

Property value

By default, it is an empty collection.

Remarks

The property cannot be modified while WriteToServer is executing.

If ColumnMappings is empty when WriteToServer is executed, then the first column in the source is mapped to the first column in the destination, the second to the second, and so on. This takes place as long as the column types are convertible, there are at least as many destination columns as source columns, and any extra destination columns are nullable.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

DestinationTableName property

Gets or sets the name of the destination table on the server.

Syntax**Visual Basic**

Public Property **DestinationTableName** As String

C#

public string **DestinationTableName** { get; set; }

Property value

The default value is a null reference (Nothing in Visual Basic).

Remarks

If the value is changed while WriteToServer is executing, the change has no effect.

If the value has not been set before a call to WriteToServer, an InvalidOperationException is raised.

It is an error to set the value to null (Nothing in Visual Basic) or the empty string.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)

NotifyAfter property

Specifies the number of rows to be processed before generating a notification event.

Syntax**Visual Basic**

Public Property **NotifyAfter** As Integer

C#

```
public int NotifyAfter { get; set; }
```

Property value

An integer representing the number of rows to be processed before generating a notification event, or zero is if the property has not been set.

Remarks

Changes made to NotifyAfter, while executing WriteToServer, do not take effect until after the next notification.

Setting it less than zero is an error.

The value of NotifyAfter and BulkCopyTimeout are mutually exclusive, so the event can fire even if no rows have been sent to the database or committed.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“BulkCopyTimeout property” on page 473](#)

Close method

Closes the ULBulkCopy instance.

Syntax**Visual Basic**

Public Sub **Close()**

C#

```
public void Close();
```

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)

Dispose method

Disposes of the ULBulkCopy instance.

Syntax**Visual Basic**

```
NotOverridable Public Sub Dispose()
```

C#

```
public void Dispose();
```

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)

WriteToServer methods

Copies all rows in the supplied array of System.Data.DataRow objects to a destination table specified by the DestinationTableName field of the ULBulkCopy object.

WriteToServer(DataRow[]) method

Copies all rows in the supplied array of System.Data.DataRow objects to a destination table specified by the DestinationTableName field of the ULBulkCopy object.

Syntax**Visual Basic**

```
Public Sub WriteToServer( _  
    ByVal rows As DataRow() _  
)
```

C#

```
public void WriteToServer(  
    DataRow[] rows  
);
```

Parameters

- ◆ **rows** An array of System.Data.DataRow objects that will be copied to the destination table.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“WriteToServer methods” on page 476](#)
- ◆ [DataRow](#)
- ◆ [“DestinationTableName property” on page 474](#)

WriteToServer(DataTable) method

Copies all rows in the supplied System.Data.DataTable to a destination table specified by the DestinationTableName of the ULBulkCopy object.

Syntax**Visual Basic**

```
Public Sub WriteToServer( _  
    ByVal table As DataTable _  
)
```

C#

```
public void WriteToServer(  
    DataTable table  
);
```

Parameters

- ◆ **table** A System.Data.DataTable whose rows will be copied to the destination table.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“WriteToServer methods” on page 476](#)
- ◆ [“DestinationTableName property” on page 474](#)
- ◆ [DataTable](#)

WriteToServer(IDataReader) method

Copies all rows in the supplied System.Data.IDataReader to a destination table specified by the DestinationTableName of the ULBulkCopy object.

Syntax**Visual Basic**

```
Public Sub WriteToServer( _  
    ByVal reader As IDataReader _  
)
```

C#

```
public void WriteToServer(  
    IDataReader reader  
);
```

Parameters

- ◆ **reader** A System.Data.IDataReader whose rows will be copied to the destination table.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“WriteToServer methods” on page 476](#)
- ◆ [IDataReader](#)
- ◆ [“DestinationTableName property” on page 474](#)

WriteToServer(DataTable, DataRowState) method

Copies all rows in the supplied System.Data.DataTable with the specified row state to a destination table specified by the DestinationTableName of the ULBulkCopy object.

Syntax**Visual Basic**

```
Public Sub WriteToServer(  
    ByVal table As DataTable, _  
    ByVal rowState As DataRowState _  
)
```

C#

```
public void WriteToServer(  
    DataTable table,  
    DataRowState rowState  
);
```

Parameters

- ◆ **table** A System.Data.DataTable whose rows will be copied to the destination table.
- ◆ **rowState** A value from the System.Data.DataRowState enumeration. Only rows matching the row state are copied to the destination.

Remarks

If rowState is specified, then only those rows that have the same row state are copied.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“WriteToServer methods” on page 476](#)

- ◆ [“DestinationTableName property” on page 474](#)
- ◆ [DataTable](#)
- ◆ [DataRowState](#)

ULRowsCopied event

This event occurs every time the number of rows specified by `NotifyAfter` have been processed.

Syntax

Visual Basic

Public Event **ULRowsCopied** As ULRowsCopiedEventHandler

C#

public event ULRowsCopiedEventHandler **ULRowsCopied** ;

Remarks

The receipt of a `ULRowsCopied` event does not imply that any rows have been committed. You cannot call the `Close` method from this event.

See also

- ◆ [“ULBulkCopy class” on page 468](#)
- ◆ [“ULBulkCopy members” on page 468](#)
- ◆ [“NotifyAfter property” on page 475](#)

ULBulkCopyColumnMapping class

Defines the mapping between a column in a ULBulkCopy instance's data source and a column in the instance's destination table. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULBulkCopyColumnMapping**

C#

public sealed class **ULBulkCopyColumnMapping**

Remarks

Restrictions: The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopy class” on page 468](#)

ULBulkCopyColumnMapping members

Public constructors

Member name	Description
ULBulkCopyColumnMapping constructors	Initializes a new instance of the “ ULBulkCopyColumnMapping class ” on page 480.

Public properties

Member name	Description
DestinationColumn property	Specifies the name of the column in the destination database table being mapped to.
DestinationOrdinal property	Specifies the ordinal value of the column in the destination database table being mapped to.
SourceColumn property	Specifies the name of the column being mapped in the data source.
SourceOrdinal property	Specifies the ordinal position of the source column within the data source.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopy class” on page 468](#)

ULBulkCopyColumnMapping constructors

Initializes a new instance of the [“ULBulkCopyColumnMapping class” on page 480](#).

ULBulkCopyColumnMapping() constructor

Creates a new column mapping.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULBulkCopyColumnMapping();
```

Remarks

Restrictions: The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping constructors” on page 481](#)

ULBulkCopyColumnMapping(Int32, Int32) constructor

Creates a new column mapping, using column ordinals or names to refer to source and destination columns.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal sourceColumnOrdinal As Integer, _  
    ByVal destinationColumnOrdinal As Integer _  
)
```

C#

```
public ULBulkCopyColumnMapping(  
    int sourceColumnOrdinal,  
    int destinationColumnOrdinal  
);
```

Parameters

- ◆ **sourceColumnOrdinal** The ordinal position of the source column within the data source. The first column in a data source has ordinal position zero.

- ◆ **destinationColumnOrdinal** The ordinal position of the destination column within the destination table. The first column in a table has ordinal position zero.

Remarks

Restrictions: The `ULBulkCopyColumnMapping` class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping constructors” on page 481](#)

ULBulkCopyColumnMapping(Int32, String) constructor

Creates a new column mapping, using a column ordinal to refer to the source column and a column name to refer to the destination column.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal sourceColumnOrdinal As Integer, _  
    ByVal destinationColumn As String _  
)
```

C#

```
public ULBulkCopyColumnMapping(  
    int sourceColumnOrdinal,  
    string destinationColumn  
);
```

Parameters

- ◆ **sourceColumnOrdinal** The ordinal position of the source column within the data source. The first column in a data source has ordinal position zero.
- ◆ **destinationColumn** The name of the destination column within the destination table.

Remarks

Restrictions: The `ULBulkCopyColumnMapping` class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping constructors” on page 481](#)

ULBulkCopyColumnMapping(String, Int32) constructor

Creates a new column mapping, using a column name to refer to the source column and a column ordinal to refer to the destination the column.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal sourceColumn As String, _  
    ByVal destinationColumnOrdinal As Integer _  
)
```

C#

```
public ULBulkCopyColumnMapping(  
    string sourceColumn,  
    int destinationColumnOrdinal  
);
```

Parameters

- ◆ **sourceColumn** The name of the source column within the data source.
- ◆ **destinationColumnOrdinal** The ordinal position of the destination column within the destination table. The first column in a table has ordinal position zero.

Remarks

Restrictions: The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping constructors” on page 481](#)

ULBulkCopyColumnMapping(String, String) constructor

Creates a new column mapping, using column names to refer to source and destination columns.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal sourceColumn As String, _  
    ByVal destinationColumn As String _  
)
```

C#

```
public ULBulkCopyColumnMapping(
```

```
        string sourceColumn,  
        string destinationColumn  
    );
```

Parameters

- ◆ **sourceColumn** The name of the source column within the data source.
- ◆ **destinationColumn** The name of the destination column within the destination table.

Remarks

Restrictions: The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping constructors” on page 481](#)

DestinationColumn property

Specifies the name of the column in the destination database table being mapped to.

Syntax

Visual Basic

Public Property **DestinationColumn** As String

C#

```
public string DestinationColumn { get; set; }
```

Property value

A string specifying the name of the column in the destination table or a null reference (Nothing in Visual Basic) if the DestinationOrdinal has priority.

Remarks

The DestinationColumn and DestinationOrdinal properties are mutually exclusive. The most recently set value takes priority.

Setting the DestinationColumn property causes the DestinationOrdinal property to be set to 1. Setting the DestinationOrdinal property causes the DestinationColumn property to be set to a null reference (Nothing in Visual Basic).

It is an error to set DestinationColumn to null or the empty string.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“DestinationOrdinal property” on page 485](#)

- ◆ [“DestinationOrdinal property” on page 485](#)

DestinationOrdinal property

Specifies the ordinal value of the column in the destination database table being mapped to.

Syntax

Visual Basic

Public Property **DestinationOrdinal** As Integer

C#

```
public int DestinationOrdinal { get; set; }
```

Property value

An integer specifying the ordinal of the column being mapped to in the destination table or -1 if the property is not set.

Remarks

The DestinationColumn and DestinationOrdinal properties are mutually exclusive. The most recently set value takes priority.

Setting the DestinationColumn property causes the DestinationOrdinal property to be set to -1. Setting the DestinationOrdinal property causes the DestinationColumn property to be set to a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“DestinationColumn property” on page 484](#)
- ◆ [“DestinationColumn property” on page 484](#)

SourceColumn property

Specifies the name of the column being mapped in the data source.

Syntax

Visual Basic

Public Property **SourceColumn** As String

C#

```
public string SourceColumn { get; set; }
```

Property value

A string specifying the name of the column in the data source or a null reference (Nothing in Visual Basic) if the SourceOrdinal has priority.

Remarks

The SourceColumn and SourceOrdinal properties are mutually exclusive. The most recently set value takes priority.

Setting the SourceColumn property causes the SourceOrdinal property to be set to `â1`. Setting the SourceOrdinal property causes the SourceColumn property to be set to a null reference (Nothing in Visual Basic).

It is an error to set SourceColumn to null or the empty string.

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“SourceOrdinal property” on page 486](#)
- ◆ [“SourceOrdinal property” on page 486](#)

SourceOrdinal property

Specifies the ordinal position of the source column within the data source.

Syntax

Visual Basic

Public Property **SourceOrdinal** As Integer

C#

```
public int SourceOrdinal { get; set; }
```

Property value

An integer specifying the ordinal of the column in the data source or -1 if the property is not set.

Remarks

The SourceColumn and SourceOrdinal properties are mutually exclusive. The most recently set value takes priority.

Setting the SourceColumn property causes the SourceOrdinal property to be set to `â1`. Setting the SourceOrdinal property causes the SourceColumn property to be set to a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)
- ◆ [“ULBulkCopyColumnMapping members” on page 480](#)
- ◆ [“SourceColumn property” on page 485](#)

ULBulkCopyColumnMappingCollection class

A collection of ULBulkCopyColumnMapping objects that inherits from System.Collections.CollectionBase. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULBulkCopyColumnMappingCollection**
Inherits CollectionBase

C#

public sealed class **ULBulkCopyColumnMappingCollection** : CollectionBase

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ♦ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ♦ [“ULBulkCopyColumnMapping class” on page 480](#)

ULBulkCopyColumnMappingCollection members

Public properties

Member name	Description
Capacity (inherited from CollectionBase)	
Count (inherited from CollectionBase)	
Item property	Gets the ULBulkCopyColumnMapping object at the specified index.

Public methods

Member name	Description
Add methods	Adds the specified ULBulkCopyColumnMapping to the collection.
Clear (inherited from CollectionBase)	
Contains method	Returns whether the specified ULBulkCopyColumnMapping object exists in the collection.

Member name	Description
CopyTo method	Copies the elements of the <code>ULBulkCopyColumnMappingCollection</code> to an array of <code>ULBulkCopyColumnMapping</code> items, starting at a particular index.
GetEnumerator (inherited from <code>CollectionBase</code>)	
IndexOf method	Returns the index of the specified <code>ULBulkCopyColumnMapping</code> within the collection.
Remove method	Removes the specified <code>ULBulkCopyColumnMapping</code> element from the <code>ULBulkCopyColumnMappingCollection</code> .
RemoveAt method	Removes the mapping at the specified index from the collection.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Item property

Gets the `ULBulkCopyColumnMapping` object at the specified index.

Syntax**Visual Basic**

```
Public Readonly Property Item ( _  
    ByVal index As Integer _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping this [  
    int index  
] { get; }
```

Parameters

- ◆ **index** The zero-based index of the `ULBulkCopyColumnMapping` object to find.

Property value

An `ULBulkCopyColumnMapping` object is returned.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Add methods

Adds the specified ULBulkCopyColumnMapping to the collection.

Add(ULBulkCopyColumnMapping) method

Adds the specified ULBulkCopyColumnMapping to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal bulkCopyColumnMapping As ULBulkCopyColumnMapping _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping Add(  
    ULBulkCopyColumnMapping bulkCopyColumnMapping  
);
```

Parameters

- ◆ **bulkCopyColumnMapping** The ULBulkCopyColumnMapping object that describes the mapping to be added to the collection.

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“Add methods” on page 489](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Add(Int32, Int32) method

Creates a new ULBulkCopyColumnMapping instance using ordinals to specify both source and destination columns, and adds the mapping to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal sourceColumnOrdinal As Integer, _  
    ByVal destinationColumnOrdinal As Integer _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping Add(  
    int sourceColumnOrdinal,  
    int destinationColumnOrdinal  
);
```

Parameters

- ◆ **sourceColumnOrdinal** The ordinal position of the source column within the data source. The first column in a data source has ordinal position zero.
- ◆ **destinationColumnOrdinal** The ordinal position of the destination column within the destination table. The first column in a table has ordinal position zero.

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“Add methods” on page 489](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Add(Int32, String) method

Creates a new ULBulkCopyColumnMapping using a column ordinal to refer to the source column and a column name to refer to the destination column, and adds mapping to the collection.

Syntax**Visual Basic**

```
Public Function Add( _  
    ByVal sourceColumnOrdinal As Integer, _  
    ByVal destinationColumn As String _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping Add(  
    int sourceColumnOrdinal,  
    string destinationColumn  
);
```

Parameters

- ◆ **sourceColumnOrdinal** The ordinal position of the source column within the data source. The first column in a data source has ordinal position zero.
- ◆ **destinationColumn** The name of the destination column within the destination table.

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“Add methods” on page 489](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Add(String, Int32) method

Creates a new ULBulkCopyColumnMapping using a column name to refer to the source column and a column ordinal to refer to the destination the column, and adds the mapping to the collection.

Creates a new column mapping, using column ordinals or names to refer to source and destination columns.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal sourceColumn As String, _  
    ByVal destinationColumnOrdinal As Integer _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping Add(  
    string sourceColumn,  
    int destinationColumnOrdinal  
);
```

Parameters

- ◆ **sourceColumn** The name of the source column within the data source.
- ◆ **destinationColumnOrdinal** The ordinal position of the destination column within the destination table. The first column in a table has ordinal position zero.

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“Add methods” on page 489](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Add(String, String) method

Creates a new ULBulkCopyColumnMapping using column names to specify both source and destination columns, and adds the mapping to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal sourceColumn As String, _  
    ByVal destinationColumn As String _  
) As ULBulkCopyColumnMapping
```

C#

```
public ULBulkCopyColumnMapping Add(  
    string sourceColumn,  
    string destinationColumn  
);
```

Parameters

- ◆ **sourceColumn** The name of the source column within the data source.
- ◆ **destinationColumn** The name of the destination column within the destination table.

Remarks

Restrictions: The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“Add methods” on page 489](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Contains method

Returns whether the specified ULBulkCopyColumnMapping object exists in the collection.

Syntax

Visual Basic

```
Public Function Contains( _  
    ByVal value As ULBulkCopyColumnMapping _  
) As Boolean
```

C#

```
public bool Contains(
```

```
    ULBulkCopyColumnMapping value  
);
```

Parameters

- ◆ **value** A valid ULBulkCopyColumnMapping object.

Return value

True if the specified mapping exists in the collection; otherwise, false.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

CopyTo method

Copies the elements of the ULBulkCopyColumnMappingCollection to an array of ULBulkCopyColumnMapping items, starting at a particular index.

Syntax

Visual Basic

```
Public Sub CopyTo( _  
    ByVal array As ULBulkCopyColumnMapping(), _  
    ByVal index As Integer _  
)
```

C#

```
public void CopyTo(  
    ULBulkCopyColumnMapping[] array,  
    int index  
);
```

Parameters

- ◆ **array** The one-dimensional ULBulkCopyColumnMapping array that is the destination of the elements copied from this ULBulkCopyColumnMappingCollection. The array must have zero-based indexing.
- ◆ **index** The zero-based index in the array at which copying begins.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

IndexOf method

Returns the index of the specified ULBulkCopyColumnMapping within the collection.

Syntax

Visual Basic

```
Public Function IndexOf( _  
    ByVal value As ULBulkCopyColumnMapping _  
) As Integer
```

C#

```
public int IndexOf(  
    ULBulkCopyColumnMapping value  
);
```

Parameters

- ◆ **value** The ULBulkCopyColumnMapping object to search for.

Return value

The zero-based index of the column mapping is returned, or -1 is returned if the column mapping is not found in the collection.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)
- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

Remove method

Removes the specified ULBulkCopyColumnMapping element from the ULBulkCopyColumnMappingCollection.

Syntax

Visual Basic

```
Public Sub Remove( _  
    ByVal value As ULBulkCopyColumnMapping _  
)
```

C#

```
public void Remove(  
    ULBulkCopyColumnMapping value  
);
```

Parameters

- ◆ **value** The ULBulkCopyColumnMapping object to be removed from the collection.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)

- ◆ [“ULBulkCopyColumnMapping class” on page 480](#)

RemoveAt method

Removes the mapping at the specified index from the collection.

Syntax

Visual Basic

```
Public Sub RemoveAt( _  
    ByVal index As Integer _  
)
```

C#

```
public void RemoveAt(  
    int index  
);
```

Parameters

- ◆ **index** The zero-based index of the ULBulkCopyColumnMapping object to be removed from the collection.

See also

- ◆ [“ULBulkCopyColumnMappingCollection class” on page 487](#)
- ◆ [“ULBulkCopyColumnMappingCollection members” on page 487](#)

ULBulkCopyOptions enumeration

A bitwise flag that specifies one or more options to use with an instance of the ULBulkCopy class.

Syntax

Visual Basic

Public Enum **ULBulkCopyOptions**

C#

public enum **ULBulkCopyOptions**

Remarks

The ULBulkCopyOptions enumeration is used when you construct a ULBulkCopy instance to specify how the WriteToServer methods will behave.

Restrictions: The ULBulkCopyOptions class is not available in the .NET Compact Framework 2.0.

Members

Member name	Description	Value
Default	Specifying only this causes the default behavior to be used.	0
KeepIdentity	When specified, the source values to be copied into an identity column are preserved. By default, new identity values are generated in the destination table.	1
UseInternalTransaction	When specified, each batch of the bulk-copy operation is executed within a transaction. When not specified, transaction arenât used. If you indicate this option and also provide a ULTransaction object to the constructor, a System.ArgumentException occurs.	2

See also

- ♦ [“ULBulkCopy class” on page 468](#)

ULCommand class

Represents a pre-compiled SQL statement or query, with or without IN parameters. This object can be used to execute a statement or query efficiently multiple times. This class cannot be inherited.

Syntax

Visual Basic

```
Public NotInheritable Class ULCommand
    Inherits DbCommand
    Implements ICloneable
```

C#

```
public sealed class ULCommand : DbCommand,
    ICloneable
```

Remarks

ULCommand objects can be created directly, or with the ULConnection.CreateCommand method. This method ensures that the command has the correct transaction for executing statements on the given connection.

The ULCommand.Transaction method must be reset after the current transaction is committed or rolled back.

ULCommand features the following methods for executing commands on an UltraLite.NET database:

Method	Description
ULCommand.ExecuteNon-Query	Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.
ULCommand.ExecuteReader()	Executes a SQL SELECT statement and returns the result set in a ULDataReader. Use this method for creating read-only result sets.
ULCommand.ExecuteResultSet() ()	UL Ext.: Executes a SQL SELECT statement and returns the result set in a ULResultSet. Use this method for creating mutable result sets.
ULCommand.ExecuteScalar	Executes a SQL SELECT statement and returns a single value.
ULCommand.ExecuteTable()	UL Ext.: Retrieves a database table in a ULTable for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and the ULCommand.IndexName can be used to specify a table sorting order. The ULCommand.CommandType must be System.Data.CommandType.TableDirect.

You can reset most properties, including the ULCommand.CommandText, and reuse the ULCommand object.

For resource management reasons, it is recommended that you explicitly dispose of commands when you are done with them. In C#, you may use a using statement to automatically call the System.ComponentModel.Component.Dispose() method or explicitly call the

System.ComponentModel.Component.Dispose() method. In Visual Basic, you always explicitly call the System.ComponentModel.Component.Dispose() method.

Inherits: System.Data.Common.DbCommand

Implements: System.Data.IDbCommand, System.IDisposable

See also

- ◆ [“ULCommand members” on page 498](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“ExecuteNonQuery method” on page 522](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteScalar method” on page 528](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [Component.Dispose](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [DbCommand](#)
- ◆ [IDbCommand](#)
- ◆ [IDisposable](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULDataReader class” on page 656](#)

ULCommand members

Public constructors

Member name	Description
ULCommand constructors	Initializes a new instance of the “ULCommand class” on page 497 .

Public properties

Member name	Description
CommandText property	Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.
CommandTimeout property	This feature is not supported by UltraLite.NET.
CommandType property	Specifies the type of command to be executed.

Member name	Description
Connection property	The connection object on which to execute the ULCommand object.
DesignTimeVisible property	Indicates if the ULCommand should be visible in a customized Windows Form Designer control.
IndexName property	UL Ext.: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType is System.Data.CommandType.TableDirect.
Parameters property	Specifies the parameters for the current statement.
Plan property	UL Ext.: Returns the access plan UltraLite.NET uses to execute a query. This property is intended primarily for use during development.
Transaction property	Specifies the ULTransaction in which the ULCommand executes.
UpdatedRowSource property	Specifies how command results are applied to the DataRow when used by the Update method of the ULDataAdapter.

Public methods

Member name	Description
BeginExecuteNonQuery methods	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, given a callback procedure and state information.
BeginExecuteReader methods	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, and retrieves the result set.
Cancel method	This method is not supported in UltraLite.NET.
CreateParameter method	Provides a ULParameter object for supplying parameters to ULCommand objects.
EndExecuteNonQuery method	Finishes asynchronous execution of a SQL statement.
EndExecuteReader method	Finishes asynchronous execution of a SQL statement, returning the requested ULDataReader.
ExecuteNonQuery method	Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.
ExecuteReader methods	Executes a SQL SELECT statement and returns the result set.
ExecuteResultSet methods	UL Ext.: Executes a SQL SELECT statement and returns the result set as a ULResultSet.
ExecuteScalar method	Executes a SQL SELECT statement and returns a single value.

Member name	Description
ExecuteTable methods	UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and ULCommand.IndexName can be used to specify a table sorting order.
Prepare method	Pre-compiles and stores the SQL statement of this command.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“ExecuteNonQuery method” on page 522](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteScalar method” on page 528](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [Component.Dispose](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [DbCommand](#)
- ◆ [IDbCommand](#)
- ◆ [IDisposable](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULDataReader class” on page 656](#)

ULCommand constructors

Initializes a new instance of the [“ULCommand class” on page 497](#).

ULCommand() constructor

Initializes a ULCommand object.

Syntax**Visual Basic**

```
Public Sub New()
```

C#

```
public ULCommand();
```

Remarks

The ULCommand object needs to have the ULCommand.CommandText, ULCommand.Connection, and ULCommand.Transaction properties set before a statement can be executed.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULCommand constructors” on page 500](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“ULCommand\(String\) constructor” on page 501](#)
- ◆ [“ULCommand\(String, ULConnection\) constructor” on page 502](#)
- ◆ [“ULCommand\(String, ULConnection, ULTransaction\) constructor” on page 503](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)

ULCommand(String) constructor

Initializes a ULCommand object with the specified command text.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal cmdText As String _  
)
```

C#

```
public ULCommand(  
    string cmdText  
);
```

Parameters

- ◆ **cmdText** The text of the SQL statement or name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.

Remarks

The ULCommand object needs to have the ULCommand.Connection and ULCommand.Transaction set before a statement can be executed.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULCommand constructors” on page 500](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“ULCommand\(\) constructor” on page 500](#)

- ◆ [“ULCommand\(String, ULConnection\) constructor” on page 502](#)
- ◆ [“ULCommand\(String, ULConnection, ULTransaction\) constructor” on page 503](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)

ULCommand(String, ULConnection) constructor

Initializes a ULCommand object with the specified command text and connection.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal cmdText As String, _  
    ByVal connection As ULConnection _  
)
```

C#

```
public ULCommand(  
    string cmdText,  
    ULConnection connection  
);
```

Parameters

- ◆ **cmdText** The text of the SQL statement or name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.
- ◆ **connection** The ULConnection object representing the current connection.

Remarks

The ULCommand object may need to have the ULCommand.Transaction set before a statement can be executed.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULCommand constructors” on page 500](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“ULCommand\(\) constructor” on page 500](#)
- ◆ [“ULCommand\(String\) constructor” on page 501](#)
- ◆ [“ULCommand\(String, ULConnection, ULTransaction\) constructor” on page 503](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULConnection class” on page 542](#)

ULCommand(String, ULConnection, ULTransaction) constructor

Initializes a ULCommand object with the specified command text, connection, and transaction.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal cmdText As String, _  
    ByVal connection As ULConnection, _  
    ByVal transaction As ULTransaction _  
)
```

C#

```
public ULCommand(  
    string cmdText,  
    ULConnection connection,  
    ULTransaction transaction  
);
```

Parameters

- ◆ **cmdText** The text of the SQL statement or name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.
- ◆ **connection** The ULConnection object representing the current connection.
- ◆ **transaction** The ULTransaction in which the ULCommand executes.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULCommand constructors” on page 500](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [“ULCommand\(\) constructor” on page 500](#)
- ◆ [“ULCommand\(String\) constructor” on page 501](#)
- ◆ [“ULCommand\(String, ULConnection\) constructor” on page 502](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULTransaction class” on page 939](#)

CommandText property

Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType is System.Data.CommandType.TableDirect. For parameterized statements, use a question mark (?) placeholder to pass parameters.

Syntax

Visual Basic

Public Overrides Property **CommandText** As String

C#

public override string **CommandText** { get; set; }

Property value

A string specifying the text of the SQL statement or the name of the table. The default is an empty string (invalid command).

Remarks

It is recommended that SELECT statements used to create read-only result sets (ULCommand.ExecuteReader() or ULCommand.ExecuteScalar) should end with " FOR READ ONLY". For some statements that use temporary tables, there may be a significant performance improvement.

Example

The following example demonstrates the use of the parameterized placeholder:

```
' Visual Basic
myCmd.CommandText = "SELECT * FROM Customers WHERE CustomerID = ?"

// C#
myCmd.CommandText = "SELECT * FROM Customers WHERE CustomerID = ?";
```

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteNonQuery method” on page 522](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteScalar method” on page 528](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)

CommandTimeout property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

Public Overrides Property **CommandTimeout** As Integer

C#

```
public override int CommandTimeout { get; set; }
```

Property value

The value is always zero.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)

CommandType property

Specifies the type of command to be executed.

Syntax**Visual Basic**

Public Overrides Property **CommandType** As CommandType

C#

```
public override CommandType CommandType { get; set; }
```

Property value

One of the System.Data.CommandType values. The default is System.Data.CommandType.Text.

Remarks

Supported command types are as follows:

- ◆ System.Data.CommandType.TableDirect - **UL Ext.:** When you specify this CommandType, the ULCommand.CommandText must be the name of a database table. You can also specify the index used to open (sort) the table with ULCommand.IndexName. Use ULCommand.ExecuteTable() or ULCommand.ExecuteReader() to access the table.
- ◆ System.Data.CommandType.Text - When you specify this CommandType, the ULCommand.CommandText must be a SQL statement or query. Use ULCommand.ExecuteNonQuery() to execute a non-query SQL statement and use either ULCommand.ExecuteReader() or ULCommand.ExecuteScalar to execute a query.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [CommandType](#)
- ◆ [CommandType.Text](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)

- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [CommandType.Text](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ExecuteNonQuery method” on page 522](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“ExecuteScalar method” on page 528](#)

Connection property

The connection object on which to execute the ULCommand object.

Syntax

Visual Basic

Public Property **Connection** As ULConnection

C#

public ULConnection **Connection** { get; set; }

Property value

The ULConnection object on which to execute the command.

Remarks

ULCommand objects must have an open connection before they can be executed.

The default is a null reference (Nothing in Visual Basic).

This is the strongly-typed version of System.Data.IDbCommand.Connection and System.Data.Common.DbCommand.Connection.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [IDbCommand.Connection](#)
- ◆ [DbCommand.Connection](#)

DesignTimeVisible property

Indicates if the ULCommand should be visible in a customized Windows Form Designer control.

Syntax

Visual Basic

Public Overrides Property **DesignTimeVisible** As Boolean

C#

```
public override bool DesignTimeVisible { get; set; }
```

Property value

True if this ULCommand instance should be visible, false if this instance should not be visible. The default is false.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)

IndexName property

UL Ext.: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType is System.Data.CommandType.TableDirect.

Syntax**Visual Basic**

```
Public Property IndexName As String
```

C#

```
public string IndexName { get; set; }
```

Property value

A string specifying the name of the index. The default is a null reference (Nothing in Visual Basic), meaning the table is opened with its primary key.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)

Parameters property

Specifies the parameters for the current statement.

Syntax**Visual Basic**

```
Public Readonly Property Parameters As ULParameterCollection
```

C#

```
public ULParameterCollection Parameters { get;}
```

Property value

A ULParameterCollection holding the parameters of the SQL statement. The default value is the empty collection.

Remarks

Use question marks in ULCommand.CommandText to indicate parameters. The parameters in the collection are specified in the same order as the question mark placeholders. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in this collection.

This is the strongly-typed version of System.Data.IDbCommand.Parameters and System.Data.Common.DbCommand.Parameters.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [IDbCommand.Connection](#)
- ◆ [DbCommand.Connection](#)
- ◆ [“CommandText property” on page 503](#)

Plan property

UL Ext.: Returns the access plan UltraLite.NET uses to execute a query. This property is intended primarily for use during development.

Syntax**Visual Basic**

```
Public Readonly Property Plan As String
```

C#

```
public string Plan { get;}
```

Property value

A string containing the text-based description of the query execution plan.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)

Transaction property

Specifies the ULTransaction in which the ULCommand executes.

Syntax

Visual Basic

Public Property **Transaction** As ULTransaction

C#

public ULTransaction **Transaction** { get; set; }

Property value

The ULTransaction in which the ULCommand executes. This should be the current transaction of the connection specified by the ULCommand.Connection. The default is a null reference (Nothing in Visual Basic).

Remarks

If a command is reused after a transaction has been committed or rolled back, this property needs to be reset.

This is the strongly-typed version of System.Data.IDbCommand.Transaction and System.Data.Common.DbCommand.Transaction.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [IDbCommand.Transaction](#)
- ◆ [DbCommand.Transaction](#)

UpdatedRowSource property

Specifies how command results are applied to the DataRow when used by the Update method of the ULDataAdapter.

Syntax

Visual Basic

Public Overrides Property **UpdatedRowSource** As UpdateRowSource

C#

public override UpdateRowSource **UpdatedRowSource** { get; set; }

Property value

One of the System.Data.UpdateRowSource values. The default value is System.Data.UpdateRowSource.Both.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [UpdateRowSource](#)
- ◆ [UpdateRowSource.Both](#)

BeginExecuteNonQuery methods

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, given a callback procedure and state information.

BeginExecuteNonQuery() method

Syntax

Visual Basic

Public Function **BeginExecuteNonQuery()** As IAsyncResult

C#

public IAsyncResult **BeginExecuteNonQuery();**

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteNonQuery methods” on page 510](#)

BeginExecuteNonQuery(AsyncCallback, Object) method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, given a callback procedure and state information.

Syntax

Visual Basic

Public Function **BeginExecuteNonQuery**(_
 ByVal *callback* As AsyncCallback, _
 ByVal *stateObject* As Object _
) As IAsyncResult

C#


```
public IAsyncResult BeginExecuteNonQuery(  
    AsyncCallback callback,  
    object stateObject  
);
```

Parameters

- ◆ **callback** An System.AsyncCallback delegate that is invoked when the command's execution has completed. Pass null (Nothing in Microsoft Visual Basic) to indicate that no callback is required.
- ◆ **stateObject** A user-defined state object that is passed to the callback procedure. Retrieve this object from within the callback procedure using the System.IAsyncResult.AsyncState property.

Return value

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking EndExecuteNonQuery(IAsyncResult), which returns the number of affected rows.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteNonQuery methods” on page 510](#)
- ◆ [“EndExecuteNonQuery method” on page 516](#)
- ◆ [IAsyncResult.AsyncState](#)
- ◆ [AsyncCallback](#)
- ◆ [IAsyncResult](#)

BeginExecuteReader methods

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, and retrieves the result set.

BeginExecuteReader() method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, and retrieves the result set.

Syntax

Visual Basic

```
Public Function BeginExecuteReader() As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteReader();
```

Return value

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking EndExecuteReader(IAsyncResult), which returns an ULDataReader instance that can be used to retrieve the returned rows.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteReader methods” on page 511](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [IAsyncResult](#)
- ◆ [“ULDataReader class” on page 656](#)

BeginExecuteReader(CommandBehavior) method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, using one of the CommandBehavior values, and retrieves the result set.

Syntax

Visual Basic

```
Public Function BeginExecuteReader( _  
    ByVal cmdBehavior As CommandBehavior _  
) As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteReader(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking EndExecuteReader(IAsyncResult), which returns an ULDataReader instance that can be used to retrieve the returned rows.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteReader methods” on page 511](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [CommandBehavior](#)

- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [IAsyncResult](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [“ULDataReader class” on page 656](#)

BeginExecuteReader(AsyncCallback, Object) method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, and retrieves the result set, given a callback procedure and state information.

Syntax

Visual Basic

```
Public Function BeginExecuteReader( _  
    ByVal callback As AsyncCallback, _  
    ByVal stateObject As Object _  
) As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteReader(  
    AsyncCallback callback,  
    object stateObject  
);
```

Parameters

- ◆ **callback** An System.AsyncCallback delegate that is invoked when the command's execution has completed. Pass null (Nothing in Microsoft Visual Basic) to indicate that no callback is required.
- ◆ **stateObject** A user-defined state object that is passed to the callback procedure. Retrieve this object from within the callback procedure using the System.IAsyncResult.AsyncState property.

Return value

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking EndExecuteReader(IAsyncResult), which returns an ULDataReader instance that can be used to retrieve the returned rows.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteReader methods” on page 511](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [AsyncCallback](#)
- ◆ [IAsyncResult.AsyncState](#)
- ◆ [IAsyncResult.AsyncState](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [IAsyncResult](#)

- ◆ [“ULDataReader class” on page 656](#)

BeginExecuteReader(AsyncCallback, Object, CommandBehavior) method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand, using one of the CommandBehavior values, and retrieves the result set, given a callback procedure and state information.

Syntax

Visual Basic

```
Public Function BeginExecuteReader( _  
    ByVal callback As AsyncCallback, _  
    ByVal stateObject As Object, _  
    ByVal cmdBehavior As CommandBehavior _  
) As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteReader(  
    AsyncCallback callback,  
    object stateObject,  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **callback** A System.AsyncCallback delegate that is invoked when the command's execution has completed. Pass null (Nothing in Microsoft Visual Basic) to indicate that no callback is required.
- ◆ **stateObject** A user-defined state object that is passed to the callback procedure. Retrieve this object from within the callback procedure using the System.IAsyncResult.AsyncState property.
- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

A System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking EndExecuteReader(IAsyncResult), which returns an ULDataReader instance that can be used to retrieve the returned rows.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteReader methods” on page 511](#)
- ◆ [AsyncCallback](#)
- ◆ [IAsyncResult.AsyncState](#)
- ◆ [CommandBehavior](#)

- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [IAsyncResult](#)
- ◆ [“EndExecuteReader method” on page 519](#)
- ◆ [“ULDataReader class” on page 656](#)

Cancel method

This method is not supported in UltraLite.NET.

Syntax

Visual Basic

Public Overrides Sub **Cancel()**

C#

public override void **Cancel()**;

Remarks

This method does nothing. UltraLite.NET commands cannot be interrupted while they are executing.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)

CreateParameter method

Provides a ULParameter object for supplying parameters to ULCommand objects.

Syntax

Visual Basic

Public Function **CreateParameter()** As ULParameter

C#

public ULParameter **CreateParameter()**;

Return value

A new parameter, as a ULParameter object.

Remarks

Some SQL statements can take parameters, indicated in the text of a statement by a question mark (?). The CreateParameter method provides a ULParameter object. You can set properties on the ULParameter to specify the value for the parameter.

This is the strongly-typed version of `System.Data.IDbCommand.CreateParameter` and `System.Data.Common.DbCommand.CreateParameter`.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [IDbCommand.CreateParameter](#)
- ◆ [DbCommand.CreateParameter](#)

EndExecuteNonQuery method

Finishes asynchronous execution of a SQL statement.

Syntax

Visual Basic

```
Public Function EndExecuteNonQuery( _  
    ByVal asyncResult As IAsyncResult _  
) As Integer
```

C#

```
public int EndExecuteNonQuery(  
    IAsyncResult asyncResult  
);
```

Parameters

- ◆ **asyncResult** The `System.IAsyncResult` returned by the call to `BeginExecuteNonQuery`.

Return value

The number of rows affected (the same behavior as `ExecuteNonQuery`).

Remarks

You must call `EndExecuteNonQuery` once for every call to `BeginExecuteNonQuery`. The call must be after `BeginExecuteNonQuery` has returned. ADO.NET is not thread safe; it is your responsibility to ensure that `BeginExecuteNonQuery` has returned. The `System.IAsyncResult` passed to `EndExecuteNonQuery` must be the same as the one returned from the `BeginExecuteNonQuery` call that is being completed. It is an error to call `EndExecuteNonQuery` to end a call to `BeginExecuteReader`, and vice versa.

If an error occurs while executing the command, the exception is thrown when `EndExecuteNonQuery` is called.

There are four ways to wait for execution to complete:

Call `EndExecuteNonQuery` Calling `EndExecuteNonQuery` blocks until the command completes. For example:

```
' Visual Basic
```

```

Dim cmd As ULCommand = new ULCommand( _
    "UPDATE Departments" _
    + " SET DepartmentName = 'Engineering'" _
    + " WHERE DepartmentID=100", _
    conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteNonQuery()
' perform other work
' this will block until the command completes
Dim rowCount As Integer = _
    cmd.EndExecuteNonQuery( res )

// C#
ULCommand cmd = new ULCommand(
    "UPDATE Departments"
    + " SET DepartmentName = 'Engineering'"
    + " WHERE DepartmentID=100",
    conn
);
IAsyncResult res = cmd.BeginExecuteNonQuery();
// perform other work
// this will block until the command completes
int rowCount = cmd.EndExecuteNonQuery( res );

```

Poll the IsCompleted property of the IAsyncResult You can poll the IsCompleted property of the IAsyncResult. For example:

```

' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "UPDATE Departments" _
    + " SET DepartmentName = 'Engineering'" _
    + " WHERE DepartmentID=100", _
    conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteNonQuery()
While( !res.IsCompleted )
    ' do other work
End While
' this will block until the command completes
Dim rowCount As Integer = _
    cmd.EndExecuteNonQuery( res )

// C#
ULCommand cmd = new ULCommand(
    "UPDATE Departments"
    + " SET DepartmentName = 'Engineering'"
    + " WHERE DepartmentID=100",
    conn
);
IAsyncResult res = cmd.BeginExecuteNonQuery();
while( !res.IsCompleted ) {
    // do other work
}
// this will block until the command completes
int rowCount = cmd.EndExecuteNonQuery( res );

```

Use the IAsyncResult.AsyncWaitHandle property to get a synchronization object You can use the IAsyncResult.AsyncWaitHandle property to get a synchronization object, and wait on that. For example:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "UPDATE Departments" _
    + " SET DepartmentName = 'Engineering'" _
    + " WHERE DepartmentID=100", _
    conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteNonQuery()
' perform other work
Dim wh As WaitHandle = res.AsyncWaitHandle
wh.WaitOne()
' this will not block because the command is finished
Dim rowCount As Integer = _
    cmd.EndExecuteNonQuery( res )

// C#
ULCommand cmd = new ULCommand(
    "UPDATE Departments"
    + " SET DepartmentName = 'Engineering'"
    + " WHERE DepartmentID=100",
    conn
);
IAsyncResult res = cmd.BeginExecuteNonQuery();
// perform other work
WaitHandle wh = res.AsyncWaitHandle;
wh.WaitOne();
// this will not block because the command is finished
int rowCount = cmd.EndExecuteNonQuery( res );
```

Specify a callback function when calling BeginExecuteNonQuery You can specify a callback function when calling BeginExecuteNonQuery. For example:

```
' Visual Basic
Private Sub callbackFunction(ByVal ar As IAsyncResult)
    Dim cmd As ULCommand = _
        CType(ar.AsyncState, ULCommand)
    ' this won't block since the command has completed
    Dim rowCount As Integer = _
        cmd.EndExecuteNonQuery( res )
End Sub

' elsewhere in the code
Private Sub DoStuff()
    Dim cmd As ULCommand = new ULCommand( _
        "UPDATE Departments" _
        + " SET DepartmentName = 'Engineering'" _
        + " WHERE DepartmentID=100", _
        conn _
    )
    Dim res As IAsyncResult = _
        cmd.BeginExecuteNonQuery( _
            callbackFunction, cmd _
        )
    ' perform other work. The callback function
    ' will be called when the command completes
End Sub

// C#
```



```
private void callbackFunction( IAsyncResult ar )
{
    ULCommand cmd = (ULCommand) ar.AsyncState;
    // this won't block since the command has completed
    int rowCount = cmd.EndExecuteNonQuery();
}

// elsewhere in the code
private void DoStuff()
{
    ULCommand cmd = new ULCommand(
        "UPDATE Departments"
        + " SET DepartmentName = 'Engineering'"
        + " WHERE DepartmentID=100",
        conn
    );
    IAsyncResult res = cmd.BeginExecuteNonQuery(
        callbackFunction, cmd
    );
    // perform other work. The callback function
    // will be called when the command completes
}
```

The callback function executes in a separate thread, so the usual caveats related to updating the user interface in a threaded program apply.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteNonQuery\(\) method” on page 510](#)
- ◆ [IAsyncResult](#)

EndExecuteReader method

Finishes asynchronous execution of a SQL statement, returning the requested ULDataReader.

Syntax

Visual Basic

```
Public Function EndExecuteReader( _  
    ByVal asyncResult As IAsyncResult _  
) As ULDataReader
```

C#

```
public ULDataReader EndExecuteReader(  
    IAsyncResult asyncResult  
);
```

Parameters

- ◆ **asyncResult** The System.IAsyncResult returned by the call to BeginExecuteReader.

Return value

An `ULDataReader` object that can be used to retrieve the requested rows (the same behavior as `ExecuteReader`).

Remarks

You must call `EndExecuteReader` once for every call to `BeginExecuteReader`. The call must be after `BeginExecuteReader` has returned. ADO.NET is not thread safe; it is your responsibility to ensure that `BeginExecuteReader` has returned. The `System.IAsyncResult` passed to `EndExecuteReader` must be the same as the one returned from the `BeginExecuteReader` call that is being completed. It is an error to call `EndExecuteReader` to end a call to `BeginExecuteNonQuery`, and vice versa.

If an error occurs while executing the command, the exception is thrown when `EndExecuteReader` is called.

There are four ways to wait for execution to complete:

Call `EndExecuteReader` Calling `EndExecuteReader` blocks until the command completes. For example:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "SELECT * FROM Departments", conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteReader()
' perform other work
' this will block until the command completes
Dim reader As ULDataReader = _
    cmd.EndExecuteReader( res )

// C#
ULCommand cmd = new ULCommand(
    "SELECT * FROM Departments", conn
);
IAsyncResult res = cmd.BeginExecuteReader();
// perform other work
// this will block until the command completes
ULDataReader reader = cmd.EndExecuteReader( res );
```

Poll the `IsCompleted` property of the `IAsyncResult` You can poll the `IsCompleted` property of the `IAsyncResult`. For example:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "SELECT * FROM Departments", conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteReader()
While( !res.IsCompleted )
    ' do other work
End While
' this will block until the command completes
Dim reader As ULDataReader = _
    cmd.EndExecuteReader( res )

// C#
ULCommand cmd = new ULCommand(
    "SELECT * FROM Departments", conn
);
```

```
IAsyncResult res = cmd.BeginExecuteReader();
while( !res.IsCompleted ) {
    // do other work
}
// this will block until the command completes
ULDataReader reader = cmd.EndExecuteReader( res );
```

Use the `IAsyncResult.AsyncWaitHandle` property to get a synchronization object You can use the `IAsyncResult.AsyncWaitHandle` property to get a synchronization object, and wait on that. For example:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "SELECT * FROM Departments", conn _
)
Dim res As IAsyncResult res = _
    cmd.BeginExecuteReader()
' perform other work
Dim wh As WaitHandle = res.AsyncWaitHandle
wh.WaitOne()
' this will not block because the command is finished
Dim reader As ULDataReader = _
    cmd.EndExecuteReader( res )

// C#
ULCommand cmd = new ULCommand(
    "SELECT * FROM Departments", conn
);
IAsyncResult res = cmd.BeginExecuteReader();
// perform other work
WaitHandle wh = res.AsyncWaitHandle;
wh.WaitOne();
// this will not block because the command is finished
ULDataReader reader = cmd.EndExecuteReader( res );
```

Specify a callback function when calling `BeginExecuteReader` You can specify a callback function when calling `BeginExecuteReader`. For example:

```
' Visual Basic
Private Sub callbackFunction(ByVal ar As IAsyncResult)
    Dim cmd As ULCommand = _
        CType(ar.AsyncState, ULCommand)
    ' this won't block since the command has completed
    Dim reader As ULDataReader = cmd.EndExecuteReader()
End Sub

' elsewhere in the code
Private Sub DoStuff()
    Dim cmd As ULCommand = new ULCommand( _
        "SELECT * FROM Departments", conn _
    )
    Dim res As IAsyncResult = _
        cmd.BeginExecuteReader( _
            callbackFunction, cmd _
        )
    ' perform other work. The callback function
    ' will be called when the command completes
End Sub

// C#
```

```
private void callbackFunction( IAsyncResult ar )
{
    ULCommand cmd = (ULCommand) ar.AsyncState;
    // this won't block since the command has completed
    ULDataReader reader = cmd.EndExecuteReader();
}

// elsewhere in the code
private void DoStuff()
{
    ULCommand cmd = new ULCommand(
        "SELECT * FROM Departments", conn
    );
    IAsyncResult res = cmd.BeginExecuteReader(
        callbackFunction, cmd
    );
    // perform other work. The callback function
    // will be called when the command completes
}
```

The callback function executes in a separate thread, so the usual caveats related to updating the user interface in a threaded program apply.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“BeginExecuteReader\(\) method” on page 511](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [IAsyncResult](#)

ExecuteNonQuery method

Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.

Syntax

Visual Basic

Public Overrides Function **ExecuteNonQuery()** As Integer

C#

public override int **ExecuteNonQuery();**

Return value

The number of rows affected.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and ULCommand.Parameters as needed.

For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, and for rollbacks, the return value is -1.

The `ULCommand.CommandType` cannot be `System.Data.CommandType.TableDirect`.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)

ExecuteReader methods

Executes a SQL SELECT statement and returns the result set.

ExecuteReader() method

Executes a SQL SELECT statement and returns the result set.

Syntax

Visual Basic

Public Function **ExecuteReader()** As ULDataReader

C#

```
public ULDataReader ExecuteReader();
```

Return value

The result set as a `ULDataReader` object.

Remarks

The statement is the current `ULCommand` object, with the `ULCommand.CommandText` and any `ULCommand.Parameters` as required. The `ULDataReader` object is a read-only result set. For editable result sets, use `ULCommand.ExecuteResultSet()`, `ULCommand.ExecuteTable()`, or a `ULDataAdapter`.

If the `ULCommand.CommandType` is `System.Data.CommandType.TableDirect`, `ExecuteReader` performs an `ULCommand.ExecuteTable()` and returns a `ULTable` downcast as a `ULDataReader`.

It is recommended that SELECT statements used with this method to create read-only result sets end with FOR READ ONLY. For some statements that use temporary tables, there may be a significant performance improvement.

This is the strongly-typed version of `System.Data.IDbCommand.ExecuteReader()` and `System.Data.Common.DbCommand.ExecuteReader()`.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteReader methods” on page 523](#)
- ◆ [“ExecuteReader\(CommandBehavior\) method” on page 524](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [IDbCommand.ExecuteReader](#)
- ◆ [DbCommand.ExecuteReader](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)

ExecuteReader(CommandBehavior) method

Executes a SQL SELECT statement with the specified command behavior and returns the result set.

Syntax

Visual Basic

```
Public Function ExecuteReader( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULDataReader
```

C#

```
public ULDataReader ExecuteReader(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of `System.Data.CommandBehavior` flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the `System.Data.CommandBehavior.Default`, `System.Data.CommandBehavior.CloseConnection`, and `System.Data.CommandBehavior.SchemaOnly` flags.

Return value

The result set as a `ULDataReader` object.

Remarks

The statement is the current `ULCommand` object, with the `ULCommand.CommandText` and any `ULCommand.Parameters` as required. The `ULDataReader` object is a read-only result set. For editable result sets, use `ULCommand.ExecuteResultSet(CommandBehavior)`, `ULCommand.ExecuteTable(CommandBehavior)`, or a `ULDataAdapter`.

If the `ULCommand.CommandType` is `System.Data.CommandType.TableDirect`, `ExecuteReader` performs an `ULCommand.ExecuteTable(CommandBehavior)` and returns a `ULTable` downcast as a `ULDataReader`.

It is recommended that `SELECT` statements used with this method to create read-only result sets end with `FOR READ ONLY`. For some statements that use temporary tables, there may be a significant performance improvement.

This is the strongly-typed version of `System.Data.IDbCommand.ExecuteReader(System.Data.CommandBehavior)` and `System.Data.Common.DbCommand.ExecuteReader(System.Data.CommandBehavior)`.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteReader methods” on page 523](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ExecuteResultSet\(CommandBehavior\) method” on page 526](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [IDbCommand.ExecuteReader](#)
- ◆ [DbCommand.ExecuteReader](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)

ExecuteResultSet methods

UL Ext.: Executes a SQL SELECT statement and returns the result set as a ULResultSet.

ExecuteResultSet() method

UL Ext.: Executes a SQL SELECT statement and returns the result set as a ULResultSet.

Syntax

Visual Basic

Public Function **ExecuteResultSet()** As ULResultSet

C#

public ULResultSet **ExecuteResultSet();**

Return value

The result set as a ULResultSet object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULResultSet object is an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use ULCommand.ExecuteTable() or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable() and returns a ULTable downcast as a ULResultSet.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteResultSet methods” on page 526](#)
- ◆ [“ExecuteResultSet\(CommandBehavior\) method” on page 526](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULTable class” on page 904](#)

ExecuteResultSet(CommandBehavior) method

UL Ext.: Executes a SQL SELECT statement with the specified command behavior and returns the result set as a ULResultSet.

Syntax

Visual Basic

```
Public Function ExecuteResultSet( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULResultSet
```

C#

```
public ULResultSet ExecuteResultSet(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

The result set as a ULResultSet object.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters as required. The ULResultSet object is an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use ULCommand.ExecuteTable (CommandBehavior) or a ULDataAdapter.

If the ULCommand.CommandType is System.Data.CommandType.TableDirect, ExecuteReader performs an ULCommand.ExecuteTable(CommandBehavior) and returns a ULTable downcast as a ULResultSet.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteResultSet methods” on page 526](#)
- ◆ [“ExecuteReader\(\) method” on page 523](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)

- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)

ExecuteScalar method

Executes a SQL SELECT statement and returns a single value.

Syntax

Visual Basic

Public Overrides Function **ExecuteScalar()** As Object

C#

public override object **ExecuteScalar();**

Return value

The first column of the first row in the result set, or a null reference (Nothing in Visual Basic) if the result set is empty.

Remarks

The statement is the current `ULCommand` object, with the `ULCommand.CommandText` and any `ULCommand.Parameters` as required.

If this method is called on a query that returns multiple rows and columns, only the first column of the first row is returned.

If the `ULCommand.CommandType` is `System.Data.CommandType.TableDirect`, `ExecuteScalar` performs an `ULCommand.ExecuteTable()` and returns the first column of the first row.

It is recommended that SELECT statements used with this method to create read-only result sets end with FOR READ ONLY. For some statements that use temporary tables, there may be a significant performance improvement.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [CommandType.TableDirect](#)

ExecuteTable methods

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and ULCommand.IndexName can be used to specify a table sorting order.

ExecuteTable() method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The ULCommand.CommandText is interpreted as the name of the table and ULCommand.IndexName can be used to specify a table sorting order.

Syntax

Visual Basic

```
Public Function ExecuteTable() As ULTable
```

C#

```
public ULTable ExecuteTable();
```

Return value

The table as a ULTable object.

Remarks

The ULCommand.CommandType must be set to System.Data.CommandType.TableDirect.

If the ULCommand.IndexName is a null reference (Nothing in Visual Basic), the primary key is used to open the table. Otherwise, the table is opened using the ULCommand.IndexName value as the name of the index by which to sort.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteTable methods” on page 529](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 505](#)
- ◆ [CommandType.TableDirect](#)
- ◆ [“ULTable class” on page 904](#)

ExecuteTable(CommandBehavior) method

UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The `ULCommand.CommandText` is interpreted as the name of the table and `ULCommand.IndexName` can be used to specify a table sorting order.

Syntax

Visual Basic

```
Public Function ExecuteTable( _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **cmdBehavior** A bitwise combination of `System.Data.CommandBehavior` flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the `System.Data.CommandBehavior.Default`, `System.Data.CommandBehavior.CloseConnection`, and `System.Data.CommandBehavior.SchemaOnly` flags.

Return value

The table as a `ULTable` object.

Remarks

The `ULCommand.CommandType` must be set to `System.Data.CommandType.TableDirect`.

If the `ULCommand.IndexName` is a null reference (Nothing in Visual Basic), the primary key is used to open the table. Otherwise, the table is opened using the `ULCommand.IndexName` value as the name of the index by which to sort.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“ExecuteTable methods” on page 529](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“IndexName property” on page 507](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)
- ◆ [“CommandType property” on page 505](#)

- ◆ [CommandType.TableDirect](#)

Prepare method

Pre-compiles and stores the SQL statement of this command.

Syntax

Visual Basic

Public Overrides Sub **Prepare()**

C#

public override void **Prepare();**

Remarks

Pre-compiling statements allows for the efficient re-use of statements when just the parameter values are changed. Changing any other property on this command unprepares the statement.

UltraLite.NET does not require you to explicitly prepare statements as all unprepared commands are prepared on calls to the various Execute methods.

See also

- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULCommand members” on page 498](#)
- ◆ [“Connection property” on page 506](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CommandText property” on page 503](#)

ULCommandBuilder class

Automatically generates single-table commands used to reconcile changes made to a System.Data.DataSet with the associated database.

Syntax

Visual Basic

Public Class **ULCommandBuilder**
Inherits DbCommandBuilder

C#

```
public class ULCommandBuilder : DbCommandBuilder
```

Remarks

The ULDataAdapter does not automatically generate the SQL statements required to reconcile changes made to a System.Data.DataSet with the associated data source. However, you can create a ULCommandBuilder object to automatically generate SQL statements for single-table updates if you set the SelectCommand property of the ULDataAdapter. Then, any additional SQL statements that you do not set are generated by the ULCommandBuilder.

Inherits: System.ComponentModel.Component

Implements: System.IDisposable

Example

The following example uses the ULCommand, along with ULDataAdapter and ULConnection, to select rows from a data source. The example is passed a connection string, a query string that is a SQL SELECT statement, and a string that is the name of the database table. The example then creates a ULCommandBuilder.

```
' Visual Basic
Public Shared Function SelectULRows(ByVal connectionString As String, _
    ByVal queryString As String, ByVal tableName As String)

    Dim connection As ULConnection = New ULConnection(connectionString)
    Dim adapter As ULDataAdapter = New ULDataAdapter()

    adapter.SelectCommand = New ULCommand(queryString, connection)

    Dim builder As ULCommandBuilder = New ULCommandBuilder(adapter)

    connection.Open()

    Dim dataSet As DataSet = New DataSet()
    adapter.Fill(dataSet, tableName)

    'code to modify data in DataSet here

    'Without the ULCommandBuilder this line would fail
    adapter.Update(dataSet, tableName)

    Return dataSet
```

```

End Function

// C#
public static DataSet SelectULRows(string connectionString,
    string queryString, string tableName)
{
    using (ULConnection connection = new ULConnection(connectionString))
    {
        ULDataAdapter adapter = new ULDataAdapter();
        adapter.SelectCommand = new ULCommand(queryString, connection);
        ULCommandBuilder builder = new ULCommandBuilder(adapter);

        connection.Open();

        DataSet dataSet = new DataSet();
        adapter.Fill(dataSet, tableName);

        //code to modify data in DataSet here

        //Without the ULCommandBuilder this line would fail
        adapter.Update(dataSet, tableName);

        return dataSet;
    }
}

```

See also

- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [DataSet](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [Component](#)
- ◆ [IDisposable](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULConnection class” on page 542](#)

ULCommandBuilder members

Public constructors

Member name	Description
ULCommandBuilder constructors	Initializes a new instance of the “ULCommandBuilder class” on page 532 .

Public properties

Member name	Description
CatalogLocation (inherited from DbCommandBuilder)	
CatalogSeparator (inherited from DbCommandBuilder)	

Member name	Description
ConflictOption (inherited from DbCommandBuilder)	
DataAdapter property	Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.
QuotePrefix (inherited from DbCommandBuilder)	
QuoteSuffix (inherited from DbCommandBuilder)	
SchemaSeparator (inherited from DbCommandBuilder)	
SetAllValues (inherited from DbCommandBuilder)	

Public methods

Member name	Description
GetDeleteCommand methods	Gets the automatically generated DbCommand object required to perform deletions at the data source.
GetInsertCommand methods	Gets the automatically generated DbCommand object required to perform insertions at the data source.
GetUpdateCommand methods	Gets the automatically generated DbCommand object required to perform updates at the data source.
QuoteIdentifier (inherited from DbCommandBuilder)	
RefreshSchema (inherited from DbCommandBuilder)	
UnquoteIdentifier (inherited from DbCommandBuilder)	

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [DataSet](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [Component](#)
- ◆ [IDisposable](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULConnection class” on page 542](#)

ULCommandBuilder constructors

Initializes a new instance of the [“ULCommandBuilder class” on page 532](#).

ULCommandBuilder() constructor

Initializes a ULCommandBuilder object.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULCommandBuilder();
```

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“ULCommandBuilder constructors” on page 535](#)
- ◆ [“ULCommandBuilder\(ULDataAdapter\) constructor” on page 535](#)

ULCommandBuilder(ULDataAdapter) constructor

Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal adapter As ULDataAdapter _  
)
```

C#

```
public ULCommandBuilder(  
    ULDataAdapter adapter  
);
```

Parameters

- ◆ **adapter** A ULDataAdapter object.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“ULCommandBuilder constructors” on page 535](#)
- ◆ [“ULDataAdapter class” on page 627](#)

DataAdapter property

Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.

Syntax

Visual Basic

Public Property **DataAdapter** As ULDataAdapter

C#

public ULDataAdapter **DataAdapter** { get; set; }

Property value

A ULDataAdapter object.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“ULDataAdapter class” on page 627](#)

GetDeleteCommand methods

Gets the automatically generated DbCommand object required to perform deletions at the data source.

GetDeleteCommand(Boolean) method

Gets the automatically generated ULCommand object required to perform deletions on the database.

Syntax

Visual Basic

Public Function **GetDeleteCommand**(_
 ByVal *useColumnsForParameterNames* As Boolean _
) As ULCommand

C#

public ULCommand **GetDeleteCommand**(
 bool *useColumnsForParameterNames*
);

Parameters

- ◆ **useColumnsForParameterNames** If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Return value

The automatically generated ULCommand object required to perform deletions.

Remarks

After the SQL statement is first generated, the application must explicitly call the `DbCommandBuilder.RefreshSchema` if it changes the `ULDataAdapter.SelectCommand` in any way. Otherwise, the `GetDeleteCommand` method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the `DbDataAdapter.Update(System.Data.DataSet)` or the `GetDeleteCommand` method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetDeleteCommand methods” on page 536](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

GetDeleteCommand() method

Gets the automatically generated `ULCommand` object required to perform deletions on the database.

Syntax**Visual Basic**

Public Function **GetDeleteCommand()** As `ULCommand`

C#

public `ULCommand` **GetDeleteCommand();**

Return value

The automatically generated `ULCommand` object required to perform deletions.

Remarks

After the SQL statement is first generated, the application must explicitly call the `DbCommandBuilder.RefreshSchema` if it changes the `ULDataAdapter.SelectCommand` in any way. Otherwise, the `GetDeleteCommand` method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the `DbDataAdapter.Update(System.Data.DataSet)` or the `GetDeleteCommand` method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetDeleteCommand methods” on page 536](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)

- ◆ [DbDataAdapter.Update](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

GetInsertCommand methods

Gets the automatically generated [DbCommand](#) object required to perform insertions at the data source.

GetInsertCommand(Boolean) method

Gets the automatically generated [ULCommand](#) object required to perform insertions on the database.

Syntax

Visual Basic

```
Public Function GetInsertCommand( _  
    ByVal useColumnsForParameterNames As Boolean _  
) As ULCommand
```

C#

```
public ULCommand GetInsertCommand(  
    bool useColumnsForParameterNames  
);
```

Parameters

- ◆ **useColumnsForParameterNames** If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Return value

The automatically generated [ULCommand](#) object required to perform insertions.

Remarks

After the SQL statement is first generated, the application must explicitly call the [DbCommandBuilder.RefreshSchema](#) if it changes the [ULDataAdapter.SelectCommand](#) in any way. Otherwise, the [GetInsertCommand](#) method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the [DbDataAdapter.Update\(System.Data.DataSet\)](#) or the [GetInsertCommand](#) method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetInsertCommand methods” on page 538](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

GetInsertCommand() method

Gets the automatically generated ULCommand object required to perform insertions on the database.

Syntax

Visual Basic

Public Function **GetInsertCommand()** As ULCommand

C#

public ULCommand **GetInsertCommand();**

Return value

The automatically generated ULCommand object required to perform insertions.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema if it changes the ULDataAdapter.SelectCommand in any way. Otherwise, the GetInsertCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the DbDataAdapter.Update(System.Data.DataSet) or the GetInsertCommand method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetInsertCommand methods” on page 538](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

GetUpdateCommand methods

Gets the automatically generated DbCommand object required to perform updates at the data source.

GetUpdateCommand(Boolean) method

Gets the automatically generated ULCommand object required to perform updates on the database.

Syntax

Visual Basic

Public Function **GetUpdateCommand**(_
 ByVal *useColumnsForParameterNames* As Boolean _
) As ULCommand

C#

```
public ULCommand GetUpdateCommand(  
    bool useColumnsForParameterNames  
);
```

Parameters

- ◆ **useColumnsForParameterNames** If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Return value

The automatically generated ULCommand object required to perform updates.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema if it changes the ULDataAdapter.SelectCommand in any way. Otherwise, the GetUpdateCommand method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the DbDataAdapter.Update(System.Data.DataSet) or the GetUpdateCommand method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetUpdateCommand methods” on page 539](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

GetUpdateCommand() method

Gets the automatically generated ULCommand object required to perform updates on the database.

Syntax**Visual Basic**

Public Function **GetUpdateCommand()** As ULCommand

C#

```
public ULCommand GetUpdateCommand();
```

Return value

The automatically generated ULCommand object required to perform updates.

Remarks

After the SQL statement is first generated, the application must explicitly call the `DbCommandBuilder.RefreshSchema` if it changes the `ULDataAdapter.SelectCommand` in any way. Otherwise, the `GetUpdateCommand` method will still be using information from the previous statement, which might not be correct. The SQL statements are first generated when the application calls either the `DbDataAdapter.Update(System.Data.DataSet)` or the `GetUpdateCommand` method.

See also

- ◆ [“ULCommandBuilder class” on page 532](#)
- ◆ [“ULCommandBuilder members” on page 533](#)
- ◆ [“GetUpdateCommand methods” on page 539](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbCommandBuilder.RefreshSchema](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DbCommandBuilder.DataAdapter](#)

ULConnection class

Represents a connection to an UltraLite.NET database. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULConnection**
Inherits DbConnection

C#

public sealed class **ULConnection** : DbConnection

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set `ULDatabaseManager.RuntimeType` to the appropriate value before using any other UltraLite.NET API.

A connection to an existing database is opened using the `ULConnection.Open`.

You must open a connection before carrying out any other operation, and you must close the connection after you have finished all operations on the connection and before your application terminates. In addition, you must close all result sets and tables opened on a connection before closing the connection.

The schema of the database can be accessed using an open connection's `ULConnection.Schema`.

Implements: `System.Data.IDbConnection`, `System.IDisposable`

See also

- ◆ [“ULConnection members” on page 542](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“Schema property” on page 552](#)
- ◆ [IDbConnection](#)
- ◆ [IDisposable](#)

ULConnection members

Public constructors

Member name	Description
ULConnection constructors	Initializes a new instance of the “ULConnection class” on page 542 .

Public fields

Member name	Description
INVALID_DATABASE_ID field	UL Ext.: A database ID constant indicating that the ULConnection.DatabaseID has not been set. This field is constant and read-only.

Public properties

Member name	Description
ConnectionString property	Specifies the parameters to use for opening a connection to an UltraLite.NET database. The connection string can be supplied using a ULConnectionParms object.
ConnectionTimeout property	This feature is not supported by UltraLite.NET.
DataSource property	This feature is not supported by UltraLite.NET.
Database property	Returns the name of the database to which the connection opens.
DatabaseID property	UL Ext.: Specifies the Database ID value to be used for global autoincrement columns.
DatabaseManager property	UL Ext.: Provides access to the singleton ULDatabaseManager object.
GlobalAutoIncrementUsage property	UL Ext.: Returns the percentage of available global autoincrement values that have been used.
LastIdentity property	UL Ext.: Returns the most recent identity value used.
Schema property	UL Ext.: Provides access to the schema of the current database associated with this connection.
ServerVersion property	This feature is not supported by UltraLite.NET.
State property	Returns the current state of the connection.
SyncParms property	UL Ext.: Specifies the synchronization settings for this connection.
SyncResult property	UL Ext.: Returns the results of the last synchronization for this connection.

Public methods

Member name	Description
BeginTransaction methods	Starts a database transaction.
ChangeDatabase method	Changes the current database for an open ULConnection.

Member name	Description
ChangeEncryptionKey method	UL Ext.: Changes the database's encryption key to the specified new key.
ChangePassword method	Changes the password for the user indicated in the connection string to the supplied new password.
Close method	Closes the database connection.
CountUploadRows methods	UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.
CreateCommand method	Creates and initializes a <code>ULCommand</code> object associated with this connection and its current transaction. You can use the properties of the <code>ULCommand</code> to control its behavior.
EnlistTransaction (inherited from <code>DbConnection</code>)	
ExecuteTable methods	UL Ext.: Retrieves in a <code>ULTable</code> a database table for direct manipulation. The table is opened (sorted) using the table's primary key.
GetLastDownloadTime method	UL Ext.: Returns the time of the most recent download of the specified publication.
GetNewUUID method	UL Ext.: Generates a new UUID (<code>System.Guid</code>).
GetSchema methods	Returns schema information for the data source of this DbConnection .
GrantConnectTo method	UL Ext.: Grants access to an UltraLite database for a user ID with a specified password.
Open method	Opens a connection to a database using the previously-specified connection string.
ResetLastDownloadTime method	UL Ext.: Resets the time of the most recent download.
RevokeConnectFrom method	UL Ext.: Revokes access to an UltraLite database from the specified user ID.
RollbackPartialDownload method	UL Ext.: Rolls back outstanding changes to the database from a partial download.
StartSynchronizationDelete method	UL Ext.: Marks all subsequent deletes made by this connection for synchronization.
StopSynchronizationDelete method	UL Ext.: Prevents delete operations from being synchronized.
Synchronize methods	UL Ext.: Synchronize the database using the current <code>ULConnection.SyncParms</code> .

Public events

Member name	Description
InfoMessage event	Occurs when UltraLite.NET sends a warning or an informational message on this connection.
StateChange event	Occurs when this connection changes state.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“Schema property” on page 552](#)
- ◆ [IDbConnection](#)
- ◆ [IDisposable](#)

ULConnection constructors

Initializes a new instance of the [“ULConnection class” on page 542](#).

ULConnection() constructor

Initializes a ULConnection object. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

Public Sub **New()**

C#

public **ULConnection()**;

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set ULDatabaseManager.RuntimeType to the appropriate value before using any other UltraLite.NET API.

The ULConnection object needs to have the ULConnection.ConnectionString set before it can be opened.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ULConnection constructors” on page 545](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“ConnectionString property” on page 547](#)

ULConnection(String) constructor

Initializes a ULConnection object with the specified connection string. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal connectionString As String _  
)
```

C#

```
public ULConnection(  
    string connectionString  
);
```

Parameters

- ◆ **connectionString** An UltraLite.NET connection string. A connection string is a semicolon-separated list of keyword-value pairs.

For a list of parameters, see the [“ConnectionString property” on page 547](#).

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set ULDatabaseManager.RuntimeType to the appropriate value before using any other UltraLite.NET API.

The connection string can be supplied using a ULConnectionParms object.

Example

The following code creates and opens a connection to the existing database \UltraLite\MyDatabase.udb on a Windows CE device.

```
' Visual Basic  
Dim openParms As ULConnectionParms = New ULConnectionParms  
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"  
Dim conn As ULConnection = _  
    New ULConnection( openParms.ToString() )  
conn.Open()  
  
// C#  
ULConnectionParms openParms = new ULConnectionParms();  
openParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";  
ULConnection conn = new ULConnection( openParms.ToString() );  
conn.Open();
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ULConnection constructors” on page 545](#)
- ◆ [“Open method” on page 570](#)

- ◆ [“ULConnection\(\) constructor” on page 545](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ConnectionString property” on page 547](#)

INVALID_DATABASE_ID field

UL Ext.: A database ID constant indicating that the ULConnection.DatabaseID has not been set. This field is constant and read-only.

Syntax

Visual Basic

```
Public Shared INVALID_DATABASE_ID As Long
```

C#

```
public const long INVALID_DATABASE_ID ;
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“DatabaseID property” on page 550](#)

ConnectionString property

Specifies the parameters to use for opening a connection to an UltraLite.NET database. The connection string can be supplied using a ULConnectionParms object.

Syntax

Visual Basic

```
Public Overrides Property ConnectionString As String
```

C#

```
public override string ConnectionString { get; set; }
```

Property value

The parameters used to open this connection in the form of a semicolon-separated list of keyword-value pairs. The default is an empty string (an invalid connection string).

Remarks

UL Ext.: The parameters used by UltraLite.NET are specific to UltraLite databases and therefore the connection string is not compatible with SQL Anywhere connection strings. For a list of parameters, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

Parameter values must not contain semi-colons (;), or begin with either a single quote (') or a double quote (") character. Leading and trailing spaces in values are ignored.

By default, connections are opened with UID=DBA and PWD=sql. To make the database more secure, change the user DBA's password or create new users (using `GrantConnectTo`) and remove the DBA user (using `RevokeConnectFrom`).

Example

The following code creates and opens a connection to the existing database `\UltraLite\MyDatabase.udb` on a Windows CE device.

```
' Visual Basic
Dim openParms As ULConnectionParms = New ULConnectionParms
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"
Dim conn As ULConnection = New ULConnection
conn.ConnectionString = openParms.ToString()
conn.Open()

// C#
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";
ULConnection conn = new ULConnection();
conn.ConnectionString = openParms.ToString();
conn.Open();
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“GrantConnectTo method” on page 570](#)

ConnectionTimeout property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

Public Overrides Readonly Property **ConnectionTimeout** As Integer

C#

public override int **ConnectionTimeout** { get; }

Property value

The value is always zero.

See also

- ◆ [“ULConnection class” on page 542](#)

- ◆ [“ULConnection members” on page 542](#)

DataSource property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

Public Overrides Readonly Property **DataSource** As String

C#

```
public override string DataSource { get;}
```

Property value

The value is always the empty string.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

Database property

Returns the name of the database to which the connection opens.

Syntax

Visual Basic

Public Overrides Readonly Property **Database** As String

C#

```
public override string Database { get;}
```

Property value

A string containing the name of the database.

Remarks

On Windows CE devices, ULConnection looks in the connection string in the following order: dbn, ce_file.

On desktop machines, ULConnection looks in the connection string in the following order: dbn, nt_file.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

DatabaselD property

UL Ext.: Specifies the Database ID value to be used for global autoincrement columns.

Syntax

Visual Basic

Public Property **DatabaselD** As Long

C#

```
public long DatabaselD { get; set; }
```

Property value

The Database ID value of the current database.

Remarks

The database ID value must be in the range [0, System.UInt32.MaxValue]. A value of `ULConnection.INVALID_DATABASE_ID` is used to indicate that the database ID has not been set for the current database.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)
- ◆ [“SetDatabaseOption method” on page 654](#)
- ◆ [UInt32.MaxValue](#)
- ◆ [“INVALID_DATABASE_ID field” on page 547](#)

DatabaseManager property

UL Ext.: Provides access to the singleton `ULDatabaseManager` object.

Syntax

Visual Basic

Public Shared Readonly Property **DatabaseManager** As ULDatabaseManager

C#

```
public const ULDatabaseManager DatabaseManager { get; }
```

Property value

A reference to the singleton `ULDatabaseManager` object.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

- ◆ [“ULDatabaseManager class” on page 639](#)

GlobalAutoIncrementUsage property

UL Ext.: Returns the percentage of available global autoincrement values that have been used.

Syntax

Visual Basic

Public Readonly Property **GlobalAutoIncrementUsage** As Short

C#

```
public short GlobalAutoIncrementUsage { get;}
```

Property value

The percentage of available global autoincrement values that have been used. It is an integer in the range [0-100], inclusive.

Remarks

If the percentage approaches 100, your application should set a new value for the global database ID using `ULConnection.DatabaseID`.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“DatabaseID property” on page 550](#)

LastIdentity property

UL Ext.: Returns the most recent identity value used.

Syntax

Visual Basic

Public Readonly Property **LastIdentity** As UInt64

C#

```
public ulong LastIdentity { get;}
```

Property value

The most recently-used identity value as an unsigned long.

Remarks

The most recent identity value used. This property is equivalent to the SQL Anywhere statement:

```
SELECT @@identity
```

LastIdentity is particularly useful in the context of global autoincrement columns.

Since this property only allows you to determine the most recently assigned default value, you should retrieve this value soon after executing the insert statement to avoid spurious results.

Occasionally, a single insert statement may include more than one column of type global autoincrement. In this case, LastIdentity is one of the generated default values, but there is no reliable means to determine from which column the value is. For this reason, you should design your database and write your insert statements to avoid this situation.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

Schema property

UL Ext.: Provides access to the schema of the current database associated with this connection.

Syntax

Visual Basic

Public Readonly Property **Schema** As ULDatabaseSchema

C#

```
public ULDatabaseSchema Schema { get;}
```

Property value

A reference to the ULDatabaseSchema object representing the schema of the database on which this connection opens.

Remarks

This property is only valid while its connection is open.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ULDatabaseSchema class” on page 646](#)

ServerVersion property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

Public Overrides Readonly Property **ServerVersion** As String

C#

```
public override string ServerVersion { get; }
```

Property value

The value is always the empty string.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

State property

Returns the current state of the connection.

Syntax

Visual Basic

Public Overrides Readonly Property **State** As ConnectionState

C#

```
public override ConnectionState State { get; }
```

Property value

System.Data.ConnectionState.Open if the connection is open, System.Data.ConnectionState.Closed if the connection is closed.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“StateChange event” on page 576](#)
- ◆ [ConnectionState.Open](#)
- ◆ [ConnectionState.Closed](#)

SyncParms property

UL Ext.: Specifies the synchronization settings for this connection.

Syntax

Visual Basic

Public Readonly Property **SyncParms** As ULSyncParms

C#

```
public ULSyncParms SyncParms { get;}
```

Property value

A reference to the ULSyncParms object representing the parameters used for synchronization by this connection. Modifications to the parameters affect the next synchronization made over this connection.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“SyncResult property” on page 554](#)
- ◆ [“ULSyncParms class” on page 868](#)

SyncResult property

UL Ext.: Returns the results of the last synchronization for this connection.

Syntax

Visual Basic

Public Readonly Property **SyncResult** As ULSyncResult

C#

```
public ULSyncResult SyncResult { get;}
```

Property value

A reference to the ULSyncResult object representing the results of the last synchronization for this connection.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“SyncParms property” on page 553](#)
- ◆ [“SyncResult property” on page 554](#)

BeginTransaction methods

Starts a database transaction.

BeginTransaction() method

Returns a transaction object. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with `ULTransaction.Commit` or `ULTransaction.Rollback`.

Syntax

Visual Basic

Public Function **BeginTransaction()** As ULTransaction

C#

public ULTransaction **BeginTransaction();**

Return value

A `ULTransaction` object representing the new transaction.

Remarks

To associate a command with a transaction object, use the `ULCommand.Transaction` property. The current transaction is automatically associated to commands created by `ULConnection.CreateCommand`.

By default, the connection does not use transactions and all commands are automatically committed as they are executed. Once the current transaction is committed or rolled back, the connection reverts to auto commit mode until the next call to `BeginTransaction`.

This is the strongly-typed version of `System.Data.IDbConnection.BeginTransaction()` and `System.Data.Common.DbConnection.BeginTransaction()`.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“BeginTransaction methods” on page 554](#)
- ◆ [“BeginTransaction\(IsolationLevel\) method” on page 555](#)
- ◆ [“Commit method” on page 941](#)
- ◆ [“Rollback method” on page 942](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [IDbConnection.BeginTransaction](#)
- ◆ [DbConnection.BeginTransaction](#)

BeginTransaction(IsolationLevel) method

Returns a transaction object with the specified isolation level. Commands associated with a transaction object are executed as a single transaction. The transaction is terminated with `ULTransaction.Commit` or `ULTransaction.Rollback`.

Syntax

Visual Basic

```
Public Function BeginTransaction( _  
    ByVal isolationLevel As IsolationLevel _  
) As ULTransaction
```

C#

```
public ULTransaction BeginTransaction(  
    IsolationLevel isolationLevel  
);
```

Parameters

- ◆ **isolationLevel** The required isolation level for the transaction. UltraLite.NET only supports System.Data.IsolationLevel.ReadUncommitted.

Return value

A ULTransaction object representing the new transaction.

Remarks

To associate a command with a transaction object, use the ULCommand.Transaction property. The current transaction is automatically associated to commands created by ULConnection.CreateCommand.

By default, the connection does not use transactions and all commands are automatically committed as they are executed. Once the current transaction is committed or rolled back, the connection reverts to auto commit mode until the next call to BeginTransaction.

This is the strongly-typed version of System.Data.IDbConnection.BeginTransaction (System.Data.IsolationLevel) and System.Data.Common.DbConnection.BeginTransaction (System.Data.IsolationLevel).

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“BeginTransaction methods” on page 554](#)
- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [“Commit method” on page 941](#)
- ◆ [“Rollback method” on page 942](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [“CreateCommand method” on page 560](#)
- ◆ [IDbConnection.BeginTransaction](#)
- ◆ [DbConnection.BeginTransaction](#)
- ◆ [IsolationLevel.ReadUncommitted](#)

ChangeDatabase method

Changes the current database for an open ULConnection.

Syntax**Visual Basic**

```
Public Overrides Sub ChangeDatabase( _  
    ByVal connectionString As String _  
)
```

C#

```
public override void ChangeDatabase(  
    string connectionString  
);
```

Parameters

- ♦ **connectionString** A complete connection string to open the connection to a new database.

Remarks

The connection to the current database is closed even if there are parameter errors.

UL Ext.: *connectionString* is a full connection string, not a dbn or dbf.

See also

- ♦ [“ULConnection class” on page 542](#)
- ♦ [“ULConnection members” on page 542](#)
- ♦ [“ConnectionString property” on page 547](#)

ChangeEncryptionKey method

UL Ext.: Changes the database's encryption key to the specified new key.

Syntax**Visual Basic**

```
Public Sub ChangeEncryptionKey( _  
    ByVal newKey As String _  
)
```

C#

```
public void ChangeEncryptionKey(  
    string newKey  
);
```

Parameters

- ♦ **newKey** The new encryption key for the database.

Remarks

If the encryption key is lost, it is not possible to open the database.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“EncryptionKey property” on page 586](#)

ChangePassword method

Changes the password for the user indicated in the connection string to the supplied new password.

Syntax**Visual Basic**

```
Public Shared Sub ChangePassword( _  
    ByVal connectionString As String, _  
    ByVal newPassword As String _  
)
```

C#

```
public static void ChangePassword(  
    string connectionString,  
    string newPassword  
);
```

Parameters

- ◆ **connectionString** The connection string that contains enough information to connect to the database that you want. The connection string may contain the user ID and the current password.
- ◆ **newPassword** The new password to set.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

Close method

Closes the database connection.

Syntax**Visual Basic**

```
Public Overrides Sub Close()
```

C#

```
public override void Close();
```


Remarks

The Close method rolls back any pending transactions and then closes the connection. An application can call Close multiple times.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)

CountUploadRows methods

UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.

CountUploadRows(Int32, UInt32) method

UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.

Syntax

Visual Basic

```
Public Function CountUploadRows( _  
    ByVal mask As Integer, _  
    ByVal threshold As UInt32 _  
) As UInt32
```

C#

```
public uint CountUploadRows(  
    int mask,  
    uint threshold  
);
```

Parameters

- ◆ **mask** The set of publications to check for rows. For more information, see the [“ULPublicationSchema class” on page 788](#).
- ◆ **threshold** The maximum number of rows to count, limiting the amount of time taken by CountUploadRows. A value of 0 corresponds to the maximum limit. A value of 1 determines if any rows need to be synchronized.

Return value

The number of rows that need to be uploaded from the specified publication(s).

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“CountUploadRows methods” on page 559](#)
- ◆ [“CountUploadRows\(Int32, Int64\) method” on page 560](#)

CountUploadRows(Int32, Int64) method

UL Ext.: Returns the number of rows that need to be uploaded when the next synchronization takes place.

Syntax

Visual Basic

```
Public Function CountUploadRows( _  
    ByVal mask As Integer, _  
    ByVal threshold As Long _  
) As Long
```

C#

```
public long CountUploadRows(  
    int mask,  
    long threshold  
);
```

Parameters

- ◆ **mask** The set of publications to check for rows. For more information, see the [“ULPublicationSchema class” on page 788](#).
- ◆ **threshold** The maximum number of rows to count, limiting the amount of time taken by CountUploadRows. A value of 0 corresponds to the maximum limit. A value of 1 determines if any rows need to be synchronized. The threshold value must be in the range [0,0xffffffff].

Return value

The number of rows that need to be uploaded from the specified publication(s).

Remarks

This method is provided for languages that do not support the System.UInt32 type natively. Use the other form of this method if your application supports it.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“CountUploadRows methods” on page 559](#)
- ◆ [“CountUploadRows\(Int32, UInt32\) method” on page 559](#)

CreateCommand method

Creates and initializes a ULCommand object associated with this connection and its current transaction. You can use the properties of the ULCommand to control its behavior.

Syntax

Visual Basic

```
Public Function CreateCommand() As ULCommand
```

C#

```
public ULCommand CreateCommand();
```

Return value

A new ULCommand object.

Remarks

You must set the ULCommand.CommandText before the command can be executed.

This is the strongly-typed version of System.Data.IDbConnection.CreateCommand and System.Data.Common.DbConnection.CreateCommand().

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [IDbConnection.CreateCommand](#)
- ◆ [DbConnection.CreateCommand](#)

ExecuteTable methods

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The table is opened (sorted) using the table's primary key.

ExecuteTable(String) method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The table is opened (sorted) using the table's primary key.

Syntax**Visual Basic**

```
Public Function ExecuteTable( _  
    ByVal tableName As String _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    string tableName  
);
```

Parameters

- ◆ **tableName** The name of the table to open.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the ULCommand.ExecuteTable() method that does not require a ULCommand instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces iAnywhere.UltraLite.Connection.GetTable() and iAnywhere.UltraLite.Table.Open()).

Example

The following code opens the table named MyTable using the table's primary key. It assumes an open ULConnection instance called conn.

```
' Visual Basic
Dim t As ULTable = conn.ExecuteTable("MyTable")
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
' cmd.CommandText = "MyTable"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable = cmd.ExecuteTable()
' cmd.Dispose()

// C#
ULTable t = conn.ExecuteTable("MyTable");
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable();
// }
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ExecuteTable methods” on page 561](#)
- ◆ [“ExecuteTable\(String, String\) method” on page 562](#)
- ◆ [“ExecuteTable\(String, String, CommandBehavior\) method” on page 564](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULCommand class” on page 497](#)

ExecuteTable(String, String) method

UL Ext.: Retrieves in a ULTable a database table for direct manipulation. The table is opened (sorted) using the specified index.

Syntax

Visual Basic

```
Public Function ExecuteTable( _  
    ByVal tableName As String, _  
    ByVal indexName As String _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    string tableName,  
    string indexName  
);
```

Parameters

- ◆ **tableName** The name of the table to open.
- ◆ **indexName** The name of the index with which to open (sort) the table.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the ULCommand.ExecuteTable() method that does not require a ULCommand instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces iAnywhere.UltraLite.Connection.GetTable() and iAnywhere.UltraLite.Table.Open()).

Example

The following code opens the table named MyTable using the index named MyIndex. It assumes an open ULConnection instance called conn.

```
' Visual Basic  
Dim t As ULTable = conn.ExecuteTable("MyTable", "MyIndex")  
' The line above is equivalent to  
' Dim cmd As ULCommand = conn.CreateCommand()  
' cmd.CommandText = "MyTable"  
' cmd.IndexName = "MyIndex"  
' cmd.CommandType = CommandType.TableDirect  
' Dim t As ULTable = cmd.ExecuteTable()  
' cmd.Dispose()  
  
// C#  
ULTable t = conn.ExecuteTable("MyTable", "MyIndex");  
// The line above is equivalent to  
// ULTable t;  
// using(ULCommand cmd = conn.CreateCommand())  
// {  
//     cmd.CommandText = "MyTable";  
//     cmd.IndexName = "MyIndex";  
//     cmd.CommandType = CommandType.TableDirect;  
//     t = cmd.ExecuteTable();  
// }
```

See also

- ◆ [“ULConnection class” on page 542](#)

- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ExecuteTable methods” on page 561](#)
- ◆ [“ExecuteTable\(String\) method” on page 561](#)
- ◆ [“ExecuteTable\(String, String, CommandBehavior\) method” on page 564](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULCommand class” on page 497](#)

ExecuteTable(String, String, CommandBehavior) method

UL Ext.: Retrieves, with the specified command behavior, a database table for direct manipulation. The table is opened (sorted) using the specified index.

Syntax

Visual Basic

```
Public Function ExecuteTable( _  
    ByVal tableName As String, _  
    ByVal indexName As String, _  
    ByVal cmdBehavior As CommandBehavior _  
) As ULTable
```

C#

```
public ULTable ExecuteTable(  
    string tableName,  
    string indexName,  
    CommandBehavior cmdBehavior  
);
```

Parameters

- ◆ **tableName** The name of the table to open.
- ◆ **indexName** The name of the index with which to open (sort) the table.
- ◆ **cmdBehavior** A bitwise combination of System.Data.CommandBehavior flags describing the results of the query and its effect on the connection. UltraLite.NET respects only the System.Data.CommandBehavior.Default, System.Data.CommandBehavior.CloseConnection, and System.Data.CommandBehavior.SchemaOnly flags.

Return value

The table as a ULTable object.

Remarks

This method is a shortcut for the ULCommand.ExecuteTable(System.Data.CommandBehavior) that does not require a ULCommand instance. It is provided to help users porting from earlier versions of UltraLite.NET (it replaces iAnywhere.UltraLite.Connection.GetTable() and iAnywhere.UltraLite.Table.Open()).

Example

The following code opens the table named MyTable using the index named MyIndex. It assumes an open ULConnection instance called conn.

```
' Visual Basic
Dim t As ULTable = conn.ExecuteTable( _
    "MyTable", "MyIndex", CommandBehavior.Default _
)
' The line above is equivalent to
' Dim cmd As ULCommand = conn.CreateCommand()
' cmd.CommandText = "MyTable"
' cmd.IndexName = "MyIndex"
' cmd.CommandType = CommandType.TableDirect
' Dim t As ULTable = cmd.ExecuteTable(CommandBehavior.Default)
' cmd.Dispose()

// C#
ULTable t = conn.ExecuteTable(
    "MyTable", "MyIndex", CommandBehavior.Default
);
// The line above is equivalent to
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.IndexName = "MyIndex";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable(CommandBehavior.Default);
// }
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ExecuteTable methods” on page 561](#)
- ◆ [“ExecuteTable\(String\) method” on page 561](#)
- ◆ [“ExecuteTable\(String, String\) method” on page 562](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [“ExecuteTable\(CommandBehavior\) method” on page 530](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [CommandBehavior](#)
- ◆ [CommandBehavior.Default](#)
- ◆ [CommandBehavior.CloseConnection](#)
- ◆ [CommandBehavior.SchemaOnly](#)

GetLastDownloadTime method

UL Ext.: Returns the time of the most recent download of the specified publication.

Syntax

Visual Basic

Public Function **GetLastDownloadTime**(_

```
ByVal mask As Integer _  
) As Date
```

C#

```
public DateTime GetLastDownloadTime(  
    int mask  
);
```

Parameters

- ◆ **mask** The mask of the publication to check. For more information, see the [“ULPublicationSchema class” on page 788](#).

Return value

The timestamp of the last download.

Remarks

The parameter

mask must reference a single publication or be the special constant `ULPublicationSchema.SYNC_ALL_DB` for the time of the last download of the full database.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ResetLastDownloadTime method” on page 571](#)
- ◆ [“SYNC_ALL_DB field” on page 789](#)

GetNewUUID method

UL Ext.: Generates a new UUID (System.Guid).

Syntax

Visual Basic

Public Function **GetNewUUID()** As Guid

C#

```
public Guid GetNewUUID();
```

Return value

A new UUID as a System.Guid.

Remarks

This method is provided here because it is not included in the .NET Compact Framework.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [Guid](#)

GetSchema methods

Returns schema information for the data source of this [DbConnection](#).

GetSchema() method

Returns the list of supported schema collections.

Syntax**Visual Basic**

```
Public Overrides Function GetSchema() As DataTable
```

C#

```
public override DataTable GetSchema();
```

Remarks

For a description of the available metadata, see [“GetSchema\(String, String\[\]\) method” on page 568](#).

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GetSchema methods” on page 567](#)

GetSchema(String) method

Returns information for the specified metadata collection for this ULConnection.

Syntax**Visual Basic**

```
Public Overrides Function GetSchema(  
    ByVal collection As String  
) As DataTable
```

C#

```
public override DataTable GetSchema(  
    string collection  
);
```

Parameters

- ◆ **collection** Name of the metadata collection. If none provided, `MetaDataCollections` is used.

Remarks

For a description of the available metadata, see [“GetSchema\(String, String\[\]\) method” on page 568](#) .

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GetSchema methods” on page 567](#)
- ◆ [“ULConnection class” on page 542](#)

GetSchema(String, String[]) method

Returns schema information for the data source of this `ULConnection` and, if specified, uses the specified string for the schema name and the specified string array for the restriction values.

Syntax**Visual Basic**

```
Public Overrides Function GetSchema( _  
    ByVal collection As String, _  
    ByVal restrictions As String() _  
) As DataTable
```

C#

```
public override DataTable GetSchema(  
    string collection,  
    string [] restrictions  
);
```

Parameters

- ◆ **collection** Name of the metadata collection. If none provided, `MetaDataCollections` is used.
- ◆ **restrictions** A set of restriction values for the requested schema.

Return value

A `DataTable` that contains schema information.

Remarks

This method is used to query the database for various metadata. Each type of metadata is given a collection name, which must be passed to receive that data. The default collection name is `MetaDataCollections`.

You can query the .NET data provider to determine the list of supported schema collections by calling the `GetSchema` method with no arguments, or with the schema collection name **`MetaDataCollections`**. This will return a `DataTable` with a list of the supported schema collections (`CollectionName`), the number of restrictions that they each support (`NumberOfRestrictions`), and the number of identifier parts that they use (`NumberOfIdentifierParts`).

Collection	Metadata
Columns	Returns information on all columns in the database.
DataSourceInformation	Returns information about the database provider.
DataTypes	Returns a list of supported data types.
ForeignKeys	Returns information on all foreign keys in the database.
IndexColumns	Returns information on all index columns in the database.
Indexes	Returns information on all indexes in the database.
MetaDataCollections	Returns a list of all collection names.
Publications	Returns information on all publications in the database.
ReservedWords	Returns a list of reserved words used by UltraLite.
Restrictions	Returns information on restrictions used in GetSchema.
Tables	Returns information on all tables in the database.

These collection names are also available as read-only properties in the `ULMetaDataCollectionNames` class.

The results returned can be filtered by specifying an array of restrictions in the call to `GetSchema`.

The restrictions available with each collection can be queried by calling:

```
GetSchema( "Restrictions" )
```

If the collection requires four restrictions, then the restrictions parameter must be either `NULL`, or a string with four values.

To filter on a particular restriction, place the string to filter by in its place in the array and leave any unused places `NULL`. For example, the `Tables` collection has three restrictions: `Table`, `TableType`, `SyncType`.

To filter the `Table` collection:

```
GetSchema( "Tables", new string[] { "my_table", NULL, NULL } ) Returns information on all tables  
named my_table.
```

```
GetSchema( "Tables", new string[] { NULL, "User", NULL } ) Returns information on all user tables.
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GetSchema methods” on page 567](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULMetaDataCollectionNames class” on page 742](#)

GrantConnectTo method

UL Ext.: Grants access to an UltraLite database for a user ID with a specified password.

Syntax

Visual Basic

```
Public Sub GrantConnectTo( _  
    ByVal uid As String, _  
    ByVal pwd As String _  
)
```

C#

```
public void GrantConnectTo(  
    string uid,  
    string pwd  
);
```

Parameters

- ◆ **uid** The user ID to receive access to the database. The maximum length of the user ID is 16 characters.
- ◆ **pwd** The password to be associated with the user ID. The maximum length is 16 characters.

Remarks

If an existing user ID is specified, this function updates the password for the user. UltraLite supports a maximum of 4 users. This method is enabled only if user authentication was enabled when the connection was opened.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“UserID property” on page 587](#)
- ◆ [“Password property” on page 586](#)
- ◆ [“ConnectionString property” on page 547](#)

Open method

Opens a connection to a database using the previously-specified connection string.

Syntax

Visual Basic

```
Public Overrides Sub Open()
```

C#

```
public override void Open();
```

Remarks

You should explicitly close or dispose of the connection when you are done with it.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“ConnectionString property” on page 547](#)
- ◆ [“State property” on page 553](#)

ResetLastDownloadTime method

UL Ext.: Resets the time of the most recent download.

Syntax

Visual Basic

```
Public Sub ResetLastDownloadTime( _  
    ByVal mask As Integer _  
)
```

C#

```
public void ResetLastDownloadTime(  
    int mask  
);
```

Parameters

- ◆ **mask** The set of publications to reset. For more information, see the [“ULPublicationSchema class” on page 788](#).

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GetLastDownloadTime method” on page 565](#)

RevokeConnectFrom method

UL Ext.: Revokes access to an UltraLite database from the specified user ID.

Syntax

Visual Basic

```
Public Sub RevokeConnectFrom( _  
    ByVal uid As String _  
)
```

C#

```
public void RevokeConnectFrom(  
    string uid  
);
```

Parameters

- ◆ **uid** The user ID whose access to the database is being revoked.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“GrantConnectTo method” on page 570](#)

RollbackPartialDownload method

UL Ext.: Rolls back outstanding changes to the database from a partial download.

Syntax**Visual Basic**

```
Public Sub RollbackPartialDownload()
```

C#

```
public void RollbackPartialDownload();
```

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“KeepPartialDownload property” on page 872](#)
- ◆ [“ResumePartialDownload property” on page 876](#)

StartSynchronizationDelete method

UL Ext.: Marks all subsequent deletes made by this connection for synchronization.

Syntax**Visual Basic**

```
Public Sub StartSynchronizationDelete()
```

C#

```
public void StartSynchronizationDelete();
```

Remarks

When this function is called, all delete operations are again synchronized, causing the rows deleted from the UltraLite database to be removed from the consolidated database as well.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“StopSynchronizationDelete method” on page 573](#)
- ◆ [“Truncate method” on page 923](#)

StopSynchronizationDelete method

UL Ext.: Prevents delete operations from being synchronized.

Syntax**Visual Basic**

Public Sub **StopSynchronizationDelete()**

C#

public void **StopSynchronizationDelete();**

Remarks

This method is useful for deleting old information on an UltraLite database to save space, while not deleting this information on the consolidated database.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“StartSynchronizationDelete method” on page 572](#)

Synchronize methods

UL Ext.: Synchronize the database using the current ULConnection.SyncParms.

Synchronize() method

UL Ext.: Synchronize the database using the current ULConnection.SyncParms.

Syntax**Visual Basic**

Public Sub **Synchronize()**

C#

public void **Synchronize();**

Remarks

A detailed result status is reported in this connection's `ULConnection.SyncResult`.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“Synchronize methods” on page 573](#)
- ◆ [“Synchronize\(ULSyncProgressListener\) method” on page 574](#)
- ◆ [“SyncParms property” on page 553](#)
- ◆ [“SyncResult property” on page 554](#)

Synchronize(ULSyncProgressListener) method

UL Ext.: Synchronize the database using the current `ULConnection.SyncParms` with progress events posted to the specified listener.

Syntax**Visual Basic**

```
Public Sub Synchronize( _  
    ByVal listener As ULSyncProgressListener _  
)
```

C#

```
public void Synchronize(  
    ULSyncProgressListener listener  
);
```

Parameters

- ◆ **listener** The object that receives synchronization progress events.

Remarks

The last event posted to the listener will have a state of `ULSyncProgressState.STATE_LAST`.

Errors during synchronization will be posted as `ULSyncProgressState.STATE_ERROR` events and then thrown as `ULExceptions`.

A detailed result status will be reported in this connection's `ULConnection.SyncResult`.

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [“Synchronize methods” on page 573](#)
- ◆ [“ULSyncProgressListener interface” on page 894](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“SyncParms property” on page 553](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“SyncResult property” on page 554](#)

InfoMessage event

Occurs when UltraLite.NET sends a warning or an informational message on this connection.

Syntax

Visual Basic

Public Event **InfoMessage** As ULInfoMessageEventHandler

C#

public event ULInfoMessageEventHandler **InfoMessage** ;

Remarks

To process UltraLite.NET warnings or informational messages, you must create a ULInfoMessageEventHandler delegate and attach it to this event.

Example

The following code defines an informational message event handler.

```
' Visual Basic
Private Sub MyInfoMessageHandler( _
    obj As Object, args As ULInfoMessageEventArgs _
)
    System.Console.WriteLine( _
        "InfoMessageHandler: " + args.NativeError + ", " _
        + args.Message _
    )
End Sub

// C#
private void MyInfoMessageHandler(
    object obj, ULInfoMessageEventArgs args
)
{
    System.Console.WriteLine(
        "InfoMessageHandler: " + args.NativeError + ", "
        + args.Message
    );
}
```

The following code adds the MyInfoMessageHandler to the connection named conn.

```
' Visual Basic
AddHandler conn.InfoMessage, AddressOf MyInfoMessageHandler

// C#
conn.InfoMessage +=
    new ULInfoMessageEventHandler(MyInfoMessageHandler);
```

Event data

- ♦ **NativeError** The SQL code corresponding to the informational message or warning returned by the database.
- ♦ **Message** The informational or warning message string returned by the database.
- ♦ **Source** The name of the ADO.NET data provider returning the message.

See also

- ♦ [“ULConnection class” on page 542](#)
- ♦ [“ULConnection members” on page 542](#)
- ♦ [“ULInfoMessageEventHandler delegate” on page 741](#)

StateChange event

Occurs when this connection changes state.

Syntax

Visual Basic

Public Overrides Event **StateChange** As StateChangeEventHandler

C#

public event override StateChangeEventHandler **StateChange** ;

Remarks

To process state change messages, you must create a System.Data.StateChangeEventHandler delegate and attach it to this event.

Example

The following code defines a state change event handler.

```
' Visual Basic
Private Sub MyStateHandler( _
    obj As Object, args As StateChangeEventArgs _
)
    System.Console.WriteLine( _
        "StateHandler: " + args.OriginalState + " to " _
        + args.CurrentState _
    )
End Sub

// C#
private void MyStateHandler(
    object obj, StateChangeEventArgs args
)
{
    System.Console.WriteLine(
        "StateHandler: " + args.OriginalState + " to "
        + args.CurrentState
    )
}
```

```
    }  
    );  
}
```

The following code adds the MyStateHandler to the connection named conn.

```
' Visual Basic  
AddHandler conn.StateChange, AddressOf MyStateHandler  
  
// C#  
conn.StateChange += new StateChangeEventHandler(MyStateHandler);
```

Event data

- ◆ **CurrentState**
- ◆ **OriginalState**

See also

- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection members” on page 542](#)
- ◆ [StateChangeEventHandler](#)

ULConnectionParms class

UL Ext.: Builds a connection string for opening a connection to an UltraLite database. The frequently-used connection parameters are individual properties on the ULConnectionParms object.

Syntax

Visual Basic

Public Class **ULConnectionParms**
Inherits Component

C#

public class **ULConnectionParms** : Component

Remarks

A ULConnectionParms object is used to specify the parameters for opening a connection (ULConnection.Open) or dropping a database (ULDatabaseManager.DropDatabase).

Leading and trailing spaces are ignored in all values. Values must not contain leading or trailing spaces, or a semicolon (;), or begin with either a single quote (') or a double quote (").

When building a connection string, you need to identify the database and specify any optional connection settings. Once you have supplied all the connection parameters by setting the appropriate properties on a ULConnectionParms object, you create a connection string using the ULConnectionParms.ToString. The resulting string is used to create a new ULConnection with the ULConnection(String) constructor or set the ULConnection.ConnectionString of an existing ULConnection object.

Identifying the database

Each instance contains platform-specific paths to the database. Only the value corresponding to the executing platform is used. For example, in the code below the path \UltraLite\mydb1.udb would be used on Windows CE, while mydb2.db would be used on other platforms.

```
' Visual Basic
Dim dbName As ULConnectionParms = new ULConnectionParms
dbName.DatabaseOnCE = "\UltraLite\mydb1.udb"
dbName.DatabaseOnDesktop = "somedir\mydb2.udb"

// C#
ULConnectionParms dbName = new ULConnectionParms();
dbName.DatabaseOnCE = "\\UltraLite\\mydb1.udb";
dbName.DatabaseOnDesktop = @"somedir\mydb2.udb";
```

The recommended extension for UltraLite database files is .udb. On Windows CE devices, the default database is \UltraLiteDB\ulstore.udb. On other Windows platforms, the default database is ulstore.udb. In C#, you must escape any backslash characters in paths or use @-quoted string literals.

If you are using multiple databases, you must specify a database name for each database. For more informaton, see the [“AdditionalParms property” on page 581](#).

Optional connection settings

Depending on your application's needs and how the database was created, you might need to supply a non-default `ULConnectionParms.UserID` and `ULConnectionParms.Password`, a database `ULConnectionParms.EncryptionKey`, and the connection `ULConnectionParms.CacheSize`. If your application is using multiple connections, you should provide a unique `ULConnectionParms.ConnectionName` for each connection.

Databases are created with a single authenticated user, DBA, whose initial password is sql. By default, connections are opened using the user ID DBA and password sql. To disable the default user, use the `ULConnection.RevokeConnectFrom`. To add a user or change a user's password, use the `ULConnection.GrantConnectTo`.

If an encryption key was supplied when the database was created, all subsequent connections to the database must use the same encryption key. To change a database's encryption key, use the `ULConnection.ChangeEncryptionKey`.

For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

See also

- ◆ [“ULConnectionParms members” on page 579](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“DropDatabase method” on page 642](#)
- ◆ [“ToString method” on page 588](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection\(String\) constructor” on page 546](#)
- ◆ [“ConnectionString property” on page 547](#)
- ◆ [“UserID property” on page 587](#)
- ◆ [“Password property” on page 586](#)
- ◆ [“EncryptionKey property” on page 586](#)
- ◆ [“CacheSize property” on page 583](#)
- ◆ [“ConnectionName property” on page 584](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)
- ◆ [“GrantConnectTo method” on page 570](#)
- ◆ [“ChangeEncryptionKey method” on page 557](#)

ULConnectionParms members

Public constructors

Member name	Description
ULConnectionParms constructor	Initializes a <code>ULConnectionParms</code> instance with its default values.

Public properties

Member name	Description
AdditionalParms property	Specifies additional parameters as a semicolon-separated list of name=value pairs. These are less commonly used parameters.
CacheSize property	Specifies the size of the cache.
ConnectionString property	Specifies a name for the connection. This is only needed if you create more than one connection to the database.
DatabaseOnCE property	Specifies the path and file name of the UltraLite database on Windows CE.
DatabaseOnDesktop property	Specifies the path and file name of the UltraLite database on Windows desktop platforms.
EncryptionKey property	Specifies a key for encrypting the database.
Password property	Specifies the password for the authenticated user.
UserID property	Specifies an authenticated user for the database.

Public methods

Member name	Description
ToString method	Returns the string representation of this instance.

Public events

Member name	Description
UnusedEvent event	Unused.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“DropDatabase method” on page 642](#)
- ◆ [“ToString method” on page 588](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection\(String\) constructor” on page 546](#)
- ◆ [“ConnectionString property” on page 547](#)
- ◆ [“UserID property” on page 587](#)
- ◆ [“Password property” on page 586](#)
- ◆ [“EncryptionKey property” on page 586](#)
- ◆ [“CacheSize property” on page 583](#)
- ◆ [“ConnectionString property” on page 584](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)
- ◆ [“GrantConnectTo method” on page 570](#)

- ◆ [“ChangeEncryptionKey method” on page 557](#)

ULConnectionParms constructor

Initializes a ULConnectionParms instance with its default values.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULConnectionParms();
```

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

AdditionalParms property

Specifies additional parameters as a semicolon-separated list of name=value pairs. These are less commonly used parameters.

Syntax

Visual Basic

```
Public Property AdditionalParms As String
```

C#

```
public string AdditionalParms { get; set; }
```

Property value

A semicolon-separated list of keyword=value additional parameters. Values of the keyword=value list must conform to the rules for ULConnection.ConnectionString. The default is a null reference (Nothing in Visual Basic).

Remarks

The values for the page size and reserve size parameters are specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix m or M to indicate megabytes.

Additional parameters are:

Keyword	Description
dbn	Identifies a loaded database to which a connection needs to be made. When a database is started, it is assigned a database name, either explicitly with the dbn parameter, or by UltraLite using the base of the file name with the extension and path removed. When opening connections, UltraLite first searches for a running database with a matching dbn. If one is not found, UltraLite starts a new database using the appropriate database file name parameter (DatabaseOnCE or DatabaseOnDesktop). This parameter is required if the application (or UltraLite engine) needs to access two different databases that have the same base file name. This parameter is only used when opening a connection with <code>ULConnection.Open</code>.
ordered_table_scans	Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default. As of release 10.0.1, when using dynamic SQL in UltraLite, if order is not important for executing a query, UltraLite will access the rows directly from the database pages rather than using the primary key index. This improves performance of fetching rows. To use this optimization, the query must be read only and must scan all the rows. When rows are expected in a specific order, an ORDER BY statement should be included as part of the SQL query. However, it's possible that some applications have come to rely on the behavior that defaults to returning rows in the primary key order. In this case, users should set the <code>ordered_table_scans</code> parameter to 1 (true, yes, on) to revert to the old behavior when iterating over a table. When <code>ordered_table_scans</code> is set to 1 (true, yes, on) and the user does not specify an ORDER BY clause or if a query would not benefit from an index, UltraLite will default to using the primary key. The following parameter string ensures that UltraLite behaves as in previous releases. <pre>createParms.AdditionalParms = "ordered_table_scans=yes"</pre> The default is 0 (false, off, no). This parameter is only used when opening a connection with <code>ULConnection.Open</code>. For more information, see “UltraLite Connection String Parameters Reference” [UltraLite - Database Management and Reference].

Keyword	Description
reserve_size	Reserves file system space for storage of UltraLite persistent data. The reserve_size parameter allows you to pre-allocate the file system space required for your UltraLite database without inserting any data. Reserving file system space can improve performance slightly and also prevent out of memory failures. By default, the persistent storage file only grows when required as the application updates the database. Note that reserve_size reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead, as well as data compression, must be considered when deriving the required file system space from the amount of database data. Running the database with test data and observing the persistent store file size is recommended. The reserve_size parameter reserves space by growing the persistent store file to the given reserve size on startup, regardless of whether the file previously existed. The file is never truncated. The following parameter string ensures that the persistent store file is at least 2 MB upon startup. createParms.AdditionalParms = "reserve_size=2m" This parameter is only used when opening a connection with ULConnection.Open.
start	Specifies the location and then starts the UltraLite engine. Only supply a StartLine (START) connection parameter if you are connecting to an engine that is not currently running. The location is only required when the UltraLite engine is not in the system path. For more information on using the UltraLite engine with UltraLite.NET, see the "RuntimeType property" on page 640.

See also

- ◆ ["ULConnectionParms class" on page 578](#)
- ◆ ["ULConnectionParms members" on page 579](#)
- ◆ ["ConnectionString property" on page 547](#)
- ◆ ["DatabaseOnCE property" on page 585](#)
- ◆ ["DatabaseOnDesktop property" on page 585](#)
- ◆ ["Open method" on page 570](#)

CacheSize property

Specifies the size of the cache.

Syntax**Visual Basic**

Public Property **CacheSize** As String

C#

public string **CacheSize** { get; set; }

Property value

A string specifying the cache size. The default is a null reference (Nothing in Visual Basic) meaning the default of 16 pages is used.

Remarks

The values for the cache size are specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix of m or M to indicate megabytes.

For example, the following sets the cache size to 128 KB.

```
connParms.CacheSize = "128k"
```

The default cache size is 16 pages. Using the default page size of 4 KB, the default cache size is therefore 64 KB. The minimum cache size is platform dependent.

The default cache size is conservative. If your testing shows the need for better performance, you should increase the cache size.

Increasing the cache size beyond the size of the database itself provides no performance improvement and large cache sizes might interfere with the number of other applications you can use.

If the cache size is unspecified or improperly specified, the default size is used.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

ConnectionString property

Specifies a name for the connection. This is only needed if you create more than one connection to the database.

Syntax**Visual Basic**

Public Property **ConnectionString** As String

C#

public string **ConnectionString** { get; set; }

Property value

A string specifying the name of the connection. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

DatabaseOnCE property

Specifies the path and file name of the UltraLite database on Windows CE.

Syntax**Visual Basic**

Public Property **DatabaseOnCE** As String

C#

public string **DatabaseOnCE** { get; set; }

Property value

A string specifying the full path to the database. If the value is a null reference (Nothing in Visual Basic), the database \UltraLiteDB\ulstore.udb is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

DatabaseOnDesktop property

Specifies the path and file name of the UltraLite database on Windows desktop platforms.

Syntax**Visual Basic**

Public Property **DatabaseOnDesktop** As String

C#

public string **DatabaseOnDesktop** { get; set; }

Property value

A string specifying the absolute or relative path to the database. If the value is a null reference (Nothing in Visual Basic), the database ulstore.udb is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

EncryptionKey property

Specifies a key for encrypting the database.

Syntax**Visual Basic**

Public Property **EncryptionKey** As String

C#

```
public string EncryptionKey { get; set; }
```

Property value

A string specifying the encryption key. The default is a null reference (Nothing in Visual Basic) meaning no encryption.

Remarks

All connections must use the same key as was specified when the database was created. Lost or forgotten keys result in completely inaccessible databases.

As with all passwords, it is best to choose a key value that cannot be easily guessed. The key can be of arbitrary length, but generally the longer the key, the better, because a shorter key is easier to guess than a longer one. Using a combination of numbers, letters, and special characters decreases the chances of someone guessing the key.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)
- ◆ [“ChangeEncryptionKey method” on page 557](#)

Password property

Specifies the password for the authenticated user.

Syntax**Visual Basic**

Public Property **Password** As String

C#

```
public string Password { get; set; }
```

Property value

A string specifying a database user ID. The default is a null reference (Nothing in Visual Basic).

Remarks

Passwords are case sensitive.

When a database is created, the password for the DBA user ID is set to sql.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)
- ◆ [“UserID property” on page 587](#)

UserID property

Specifies an authenticated user for the database.

Syntax**Visual Basic**

Public Property **UserID** As String

C#

public string **UserID** { get; set; }

Property value

A string specifying a database user ID. The default value is a null reference (Nothing in Visual Basic).

Remarks

User IDs are case-insensitive.

Databases are initially created with a single authenticated user named DBA.

If both the user ID and password are not supplied, the user DBA with password sql are used. To make the database more secure, change the user DBA's password or create new users (using `ULConnection.GrantConnectTo`) and remove the DBA user (using `ULConnection.RevokeConnectFrom`).

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)
- ◆ [“Password property” on page 586](#)
- ◆ [“GrantConnectTo method” on page 570](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)

ToString method

Returns the string representation of this instance.

Syntax

Visual Basic

Public Overrides Function **ToString()** As String

C#

public override string **ToString()**;

Return value

The string representation of this instance as a semicolon-separated list keyword=value pairs.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

UnusedEvent event

Unused.

Syntax

Visual Basic

Public Event **UnusedEvent** As ULConnectionParms.UnusedEventHandler

C#

public event ULConnectionParms.UnusedEventHandler **UnusedEvent** ;

Remarks

This public Event is provided to fix a Visual Studio .NET bug relating to the integration of this class in Visual Basic .NET projects. It has no functional use.

See also

- ◆ [“ULConnectionParms class” on page 578](#)
- ◆ [“ULConnectionParms members” on page 579](#)

ULConnectionParms.UnusedEventHandler delegate

UL Ext.: Unused.

Syntax

Visual Basic

```
Public Delegate Sub ULConnectionParms.UnusedEventHandler( _  
    ByVal sender As Object, _  
    ByVal args As EventArgs _  
)
```

C#

```
public delegate void ULConnectionParms.UnusedEventHandler(  
    object sender,  
    EventArgs args  
);
```

Remarks

This public Delegate is provided to fix a Visual Studio .NET bug relating to the integration of this class in Visual Basic .NET projects. It has no functional use.

ULConnectionStringBuilder class

Builds a connection string for opening a connection to an UltraLite database. The frequently-used connection parameters are individual properties on the ULConnectionStringBuilder object. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULConnectionStringBuilder**
Inherits DbConnectionStringBuilder

C#

public sealed class **ULConnectionStringBuilder** : DbConnectionStringBuilder

Remarks

Restrictions: The ULConnectionStringBuilder class is not available in the .NET Compact Framework 2.0.

A ULConnectionStringBuilder object is used to specify the parameters for opening a connection (ULConnection.Open) or dropping a database (ULDatabaseManager.DropDatabase).

Leading and trailing spaces are ignored in all values. Values must not contain leading or trailing spaces, or a semicolon (;), or begin with either a single quote (') or a double quote (").

When building a connection string, you need to identify the database and specify any optional connection settings. Once you have supplied all the connection parameters by setting the appropriate properties on a ULConnectionStringBuilder object, you create a connection string using the System.Data.Common.DbConnectionStringBuilder.ConnectionString. The resulting string is used to create a new ULConnection with the ULConnection(String) constructor or set the ULConnection.ConnectionString of an existing ULConnection object.

Identifying the database

Each instance contains platform-specific paths to the database. Only the value corresponding to the executing platform is used. For example, in the code below the path \UltraLite\mydb1.udb would be used on Windows CE, while mydb2.db would be used on other platforms.

```
' Visual Basic
Dim dbName As ULConnectionStringBuilder = _
    new ULConnectionStringBuilder
dbName.DatabaseOnCE = "\UltraLite\mydb1.udb"
dbName.DatabaseOnDesktop = "somedir\mydb2.udb"

// C#
ULConnectionStringBuilder dbName = new ULConnectionStringBuilder();
dbName.DatabaseOnCE = "\\UltraLite\\mydb1.udb";
dbName.DatabaseOnDesktop = @"somedir\mydb2.udb";
```

The recommended extension for UltraLite database files is .udb. On Windows CE devices, the default database is \UltraLiteDB\ulstore.udb. On other Windows platforms, the default database is ulstore.udb. In C#, you must escape any backslash characters in paths or use @-quoted string literals.

If you are using multiple databases, you must specify a database name for each database. For more informaton, see the [“DatabaseName property” on page 596](#).

Optional connection settings

Depending on your application's needs and how the database was created, you might need to supply a non-default `ULConnectionStringBuilder.UserID` and `ULConnectionStringBuilder.Password`, a database `ULConnectionStringBuilder.DatabaseKey`, and the connection `ULConnectionStringBuilder.CacheSize`. If your application is using multiple connections, you should provide a unique `ULConnectionStringBuilder.ConnectionName` for each connection.

Databases are created with a single authenticated user, DBA, whose initial password is sql. By default, connections are opened using the user ID DBA and password sql. To disable the default user, use the `ULConnection.RevokeConnectFrom`. To add a user or change a user's password, use the `ULConnection.GrantConnectTo`.

If an encryption key was supplied when the database was created, all subsequent connections to the database must use the same encryption key. To change a database's encryption key, use the `ULConnection.ChangeEncryptionKey`.

For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

See also

- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“DropDatabase method” on page 642](#)
- ◆ [DbConnectionStringBuilder.ConnectionString](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection\(String\) constructor” on page 546](#)
- ◆ [“ConnectionString property” on page 547](#)
- ◆ [“DatabaseName property” on page 596](#)
- ◆ [“UserID property” on page 602](#)
- ◆ [“Password property” on page 600](#)
- ◆ [“DatabaseKey property” on page 596](#)
- ◆ [“CacheSize property” on page 595](#)
- ◆ [“ConnectionName property” on page 595](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)
- ◆ [“GrantConnectTo method” on page 570](#)
- ◆ [“ChangeEncryptionKey method” on page 557](#)

ULConnectionStringBuilder members

Public constructors

Member name	Description
ULConnectionStringBuilder constructors	Initializes a new instance of the “ULConnectionStringBuilder class” on page 590 .

Public properties

Member name	Description
BrowsableConnectionString (inherited from DbConnectionStringBuilder)	
CacheSize property	UL Ext.: Specifies the size of the cache.
ConnectionString property	Specifies a name for the connection. This is only needed if you create more than one connection to the database.
ConnectionString (inherited from DbConnectionStringBuilder)	
Count (inherited from DbConnectionStringBuilder)	
DatabaseKey property	Specifies a key for encrypting the database.
DatabaseName property	Specifies a name for the database or the name of a loaded database to which a connection needs to be made.
DatabaseOnCE property	UL Ext.: Specifies the path and file name of the UltraLite database on Windows CE.
DatabaseOnDesktop property	UL Ext.: Specifies the path and file name of the UltraLite database on Windows desktop platforms.
IsFixedSize (inherited from DbConnectionStringBuilder)	
IsReadOnly (inherited from DbConnectionStringBuilder)	
Item property	Specifies the value of the specified connection keyword.
Keys (inherited from DbConnectionStringBuilder)	
OrderedTableScans property	Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default.
Password property	Specifies the password for the authenticated user.
ReserveSize property	UL Ext.: Specifies the reserve file system space for storage of UltraLite persistent data.
StartLine property	Specifies the location and then starts the UltraLite engine.
UserID property	Specifies an authenticated user for the database.
Values (inherited from DbConnectionStringBuilder)	

Public methods

Member name	Description
Add (inherited from DbConnectionStringBuilder)	
Clear (inherited from DbConnectionStringBuilder)	
ContainsKey method	Determines whether the ULConnectionStringBuilder object contains a specific keyword.
EquivalentTo method	Compares the connection information in this ULConnectionStringBuilder object with the connection information in the supplied DbConnectionStringBuilder object.
GetShortName method	Retrieves the short version of the supplied keyword.
Remove method	Removes the entry with the specified key from the ULConnectionStringBuilder instance.
ShouldSerialize (inherited from DbConnectionStringBuilder)	
ToString (inherited from DbConnectionStringBuilder)	
TryGetValue method	Retrieves a value corresponding to the supplied key from this ULConnectionStringBuilder.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“Open method” on page 570](#)
- ◆ [“DropDatabase method” on page 642](#)
- ◆ [DbConnectionStringBuilder.ConnectionString](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“ULConnection\(String\) constructor” on page 546](#)
- ◆ [“ConnectionString property” on page 547](#)
- ◆ [“DatabaseName property” on page 596](#)
- ◆ [“UserID property” on page 602](#)
- ◆ [“Password property” on page 600](#)
- ◆ [“DatabaseKey property” on page 596](#)
- ◆ [“CacheSize property” on page 595](#)
- ◆ [“ConnectionName property” on page 595](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)
- ◆ [“GrantConnectTo method” on page 570](#)
- ◆ [“ChangeEncryptionKey method” on page 557](#)

ULConnectionStringBuilder constructors

Initializes a new instance of the [“ULConnectionStringBuilder class” on page 590](#).

ULConnectionStringBuilder() constructor

Initializes a ULConnectionStringBuilder instance with its default values.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULConnectionStringBuilder();
```

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“ULConnectionStringBuilder constructors” on page 594](#)

ULConnectionStringBuilder(String) constructor

Initializes a ULConnectionStringBuilder instance with the specified connection string.

Syntax

Visual Basic

```
Public Sub New(  
    ByVal connectionString As String _  
)
```

C#

```
public ULConnectionStringBuilder(  
    string connectionString  
);
```

Parameters

- ◆ **connectionString** An UltraLite.NET connection string. A connection string is a semicolon-separated list of keyword-value pairs.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“ULConnectionStringBuilder constructors” on page 594](#)

CacheSize property

UL Ext.: Specifies the size of the cache.

Syntax

Visual Basic

Public Property **CacheSize** As String

C#

public string **CacheSize** { get; set; }

Property value

A string specifying the cache size. The default is a null reference (Nothing in Visual Basic) meaning the default of 16 pages is used.

Remarks

The values for the cache size are specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix of m or M to indicate megabytes.

For example, the following sets the cache size to 128 KB.

```
connParms.CacheSize = "128k"
```

The default cache size is 16 pages. Using the default page size of 4 KB, the default cache size is therefore 64 KB. The minimum cache size is platform dependent.

The default cache size is conservative. If your testing shows the need for better performance, you should increase the cache size.

Increasing the cache size beyond the size of the database itself provides no performance improvement and large cache sizes might interfere with the number of other applications you can use.

If the cache size is unspecified or improperly specified, the default size is used.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

ConnectionString property

Specifies a name for the connection. This is only needed if you create more than one connection to the database.

Syntax

Visual Basic

Public Property **ConnectionString** As String

C#

```
public string ConnectionString { get; set; }
```

Property value

A string specifying the name of the connection. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

DatabaseKey property

Specifies a key for encrypting the database.

Syntax**Visual Basic**

```
Public Property DatabaseKey As String
```

C#

```
public string DatabaseKey { get; set; }
```

Property value

A string specifying the encryption key. The default is a null reference (Nothing in Visual Basic) meaning no encryption.

Remarks

All connections must use the same key as was specified when the database was created. Lost or forgotten keys result in completely inaccessible databases.

As with all passwords, it is best to choose a key value that cannot be easily guessed. The key can be of arbitrary length, but generally the longer the key, the better, because a shorter key is easier to guess than a longer one. Using a combination of numbers, letters, and special characters decreases the chances of someone guessing the key.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“ChangeEncryptionKey method” on page 557](#)

DatabaseName property

Specifies a name for the database or the name of a loaded database to which a connection needs to be made.

Syntax**Visual Basic**

Public Property **DatabaseName** As String

C#

public string **DatabaseName** { get; set; }

Property value

A string specifying the name of the database. The default is a null reference (Nothing in Visual Basic).

Remarks

When a database is started, it is assigned a database name, either explicitly with the dbn parameter, or by UltraLite using the base of the file name with the extension and path removed.

When opening connections, UltraLite first searches for a running database with a matching dbn. If one is not found, UltraLite starts a new database using the appropriate database file name parameter (DatabaseOnCE or DatabaseOnDesktop).

This parameter is required if the application (or UltraLite engine) needs to access two different databases that have the same base file name.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“DatabaseOnCE property” on page 597](#)
- ◆ [“DatabaseOnDesktop property” on page 598](#)

DatabaseOnCE property

UL Ext.: Specifies the path and file name of the UltraLite database on Windows CE.

Syntax**Visual Basic**

Public Property **DatabaseOnCE** As String

C#

public string **DatabaseOnCE** { get; set; }

Property value

A string specifying the full path to the database. If the value is a null reference (Nothing in Visual Basic), the database \UltraLiteDB\ulstore.udb is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)

- ◆ [“ULConnectionStringBuilder members” on page 591](#)

DatabaseOnDesktop property

UL Ext.: Specifies the path and file name of the UltraLite database on Windows desktop platforms.

Syntax

Visual Basic

Public Property **DatabaseOnDesktop** As String

C#

```
public string DatabaseOnDesktop { get; set; }
```

Property value

A string specifying the absolute or relative path to the database. If the value is a null reference (Nothing in Visual Basic), the database `ulstore.udb` is used. In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

Item property

Specifies the value of the specified connection keyword.

Syntax

Visual Basic

```
Public Overrides Default Property Item ( _  
    ByVal keyword As String _  
) As Object
```

C#

```
public override object this [  
    string keyword  
] { get; set; }
```

Parameters

- ◆ **keyword** The name of the connection keyword.

Property value

An object representing the value of the specified connection keyword.

Remarks

Connection keywords and the corresponding properties on ULConnectionStringBuilder are described in the table below:

Keyword	Corresponding Property
cache_size	“CacheSize property” on page 595
ce_file	“DatabaseOnCE property” on page 597
con	“ConnectionName property” on page 595
dbkey	“DatabaseKey property” on page 596
dbn	“DatabaseName property” on page 596
nt_file	“DatabaseOnDesktop property” on page 598
pwd	“Password property” on page 600
reserve_size	“ReserveSize property” on page 601
start	“StartLine property” on page 601
uid	“UserID property” on page 602

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“CacheSize property” on page 595](#)
- ◆ [“DatabaseOnCE property” on page 597](#)
- ◆ [“ConnectionName property” on page 595](#)
- ◆ [“DatabaseKey property” on page 596](#)
- ◆ [“DatabaseName property” on page 596](#)
- ◆ [“DatabaseOnDesktop property” on page 598](#)
- ◆ [“Password property” on page 600](#)
- ◆ [“ReserveSize property” on page 601](#)
- ◆ [“StartLine property” on page 601](#)
- ◆ [“UserID property” on page 602](#)

OrderedTableScans property

Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default.

Syntax

Visual Basic

Public Property **OrderedTableScans** As String

C#

```
public string OrderedTableScans { get; set; }
```

Property value

A boolean string specifying whether to use ordered table scans or not. For example, true/false, yes/no, 1/0, and so on. The default value is a null reference (Nothing in Visual Basic).

Remarks

As of release 10.0.1, when using dynamic SQL in UltraLite, if order is not important for executing a query, UltraLite will access the rows directly from the database pages rather than using the primary key index. This improves performance of fetching rows. To use this optimization, the query must be read only and must scan all the rows.

When rows are expected in a specific order, an ORDER BY statement should be included as part of the SQL query. However, it's possible that some applications have come to rely on the behavior that defaults to returning rows in the primary key order. In this case, users should set the OrderedTableScans parameter to 1 (true, yes, on) to revert to the old behavior when iterating over a table.

When OrderedTableScans is set to 1 (true, yes, on) and the user does not specify an ORDER BY clause or if a query would not benefit from an index, UltraLite will default to using the primary key.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

Password property

Specifies the password for the authenticated user.

Syntax**Visual Basic**

```
Public Property Password As String
```

C#

```
public string Password { get; set; }
```

Property value

A string specifying a database user ID. The default is a null reference (Nothing in Visual Basic).

Remarks

Passwords are case sensitive.

When a database is created, the password for the DBA user ID is set to sql.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)

- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“UserID property” on page 602](#)

ReserveSize property

UL Ext.: Specifies the reserve file system space for storage of UltraLite persistent data.

Syntax

Visual Basic

Public Property **ReserveSize** As String

C#

```
public string ReserveSize { get; set; }
```

Property value

A string specifying the reserve size. The default is a null reference (Nothing in Visual Basic).

Remarks

The values for the reserve size parameter is specified in units of bytes. Use the suffix k or K to indicate units of kilobytes and the suffix m or M to indicate megabytes.

The `reserve_size` parameter allows you to pre-allocate the file system space required for your UltraLite database without inserting any data. Reserving file system space can improve performance slightly and also prevent out of memory failures. By default, the persistent storage file only grows when required as the application updates the database.

Note that `reserve_size` reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead, as well as data compression, must be considered when deriving the required file system space from the amount of database data. Running the database with test data and observing the persistent store file size is recommended.

The `reserve_size` parameter reserves space by growing the persistent store file to the given reserve size on startup, regardless of whether the file previously existed. The file is never truncated.

The following parameter string ensures that the persistent store file is at least 2 MB upon startup.

```
connParms.ReserveSize = "2m"
```

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

StartLine property

Specifies the location and then starts the UltraLite engine.

Syntax**Visual Basic**

Public Property **StartLine** As String

C#

public string **StartLine** { get; set; }

Property value

A string specifying the location of the UltraLite engine executable. The default value is a null reference (Nothing in Visual Basic).

Remarks

Only supply a StartLine (START) connection parameter if you are connecting to an engine that is not currently running.

For more information on using the UltraLite engine with UltraLite.NET, see the [“RuntimeType property” on page 640](#).

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“RuntimeType property” on page 640](#)

UserID property

Specifies an authenticated user for the database.

Syntax**Visual Basic**

Public Property **UserID** As String

C#

public string **UserID** { get; set; }

Property value

A string specifying a database user ID. The default value is a null reference (Nothing in Visual Basic).

Remarks

User IDs are case-insensitive.

Databases are initially created with a single authenticated user named DBA.

If both the user ID and password are not supplied, the user DBA with password sql are used. To make the database more secure, change the user DBA's password or create new users (using `ULConnection.GrantConnectTo`) and remove the DBA user (using `ULConnection.RevokeConnectFrom`).

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)
- ◆ [“Password property” on page 600](#)
- ◆ [“GrantConnectTo method” on page 570](#)
- ◆ [“RevokeConnectFrom method” on page 571](#)

ContainsKey method

Determines whether the ULConnectionStringBuilder object contains a specific keyword.

Syntax**Visual Basic**

```
Public Overrides Function ContainsKey( _  
    ByVal keyword As String _  
) As Boolean
```

C#

```
public override bool ContainsKey(  
    string keyword  
);
```

Parameters

- ◆ **keyword** The name of the connection keyword.

Return value

True if this connection string builder contains a value for the specified keyword, otherwise returns false.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

EquivalentTo method

Compares the connection information in this ULConnectionStringBuilder object with the connection information in the supplied DbConnectionStringBuilder object.

Syntax**Visual Basic**

```
Public Overrides Function EquivalentTo( _  
    ByVal connectionStringBuilder As DbConnectionStringBuilder _  
) As Boolean
```

C#

```
public override bool EquivalentTo(  
    DbConnectionStringBuilder connectionStringBuilder  
);
```

Parameters

- ♦ **connectionStringBuilder** The other DbConnectionStringBuilder object to compare this ULConnectionStringBuilder object to.

Return value

True if this object is equivalent to the specified DbConnectionStringBuilder object, otherwise returns false.

See also

- ♦ [“ULConnectionStringBuilder class” on page 590](#)
- ♦ [“ULConnectionStringBuilder members” on page 591](#)
- ♦ [DbConnectionStringBuilder](#)

GetShortName method

Retrieves the short version of the supplied keyword.

Syntax

Visual Basic

```
Public Shared Function GetShortName( _  
    ByVal keyword As String _  
) As String
```

C#

```
public static string GetShortName(  
    string keyword  
);
```

Parameters

- ♦ **keyword** The key of the item to retrieve.

Return value

The short version of the supplied keyword if keyword is recognized, null otherwise.

See also

- ♦ [“ULConnectionStringBuilder class” on page 590](#)
- ♦ [“ULConnectionStringBuilder members” on page 591](#)

Remove method

Removes the entry with the specified key from the ULConnectionStringBuilder instance.

Syntax

Visual Basic

```
Public Overrides Function Remove( _  
    ByVal keyword As String _  
) As Boolean
```

C#

```
public override bool Remove(  
    string keyword  
);
```

Parameters

- ◆ **keyword** The name of the connection keyword.

Return value

True if the key existed within the connection string and was removed; false if the key did not exist.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

TryGetValue method

Retrieves a value corresponding to the supplied key from this ULConnectionStringBuilder.

Syntax

Visual Basic

```
Public Overrides Function TryGetValue( _  
    ByVal keyword As String, _  
    ByVal value As Object _  
) As Boolean
```

C#

```
public override bool TryGetValue(  
    string keyword,  
    object value  
);
```

Parameters

- ◆ **keyword** The key of the item to retrieve.
- ◆ **value** The value corresponding to the key.

Return value

True if keyword was found within the connection string, false otherwise.

Remarks

The TryGetValue method lets developers safely retrieve a value from a ULConnectionStringBuilder without needing to first call the ContainsKey method. Because TryGetValue does not raise an exception when you call it, passing in a nonexistent key, you do not have to look for a key before retrieving its value. Calling TryGetValue with a nonexistent key will place the null value (Nothing in Visual Basic) in the value parameter.

See also

- ◆ [“ULConnectionStringBuilder class” on page 590](#)
- ◆ [“ULConnectionStringBuilder members” on page 591](#)

ULCreateParms class

UL Ext.: Builds a string of creation-time options for creating an UltraLite database.

Syntax

Visual Basic

Public Class **ULCreateParms**

C#

public class **ULCreateParms**

Remarks

A ULCreateParms object is used to specify the parameters for creating a database (ULDatabaseManager.CreateDatabase(string,byte[],string)).

Leading and trailing spaces are ignored in all string values. Values must not contain leading or trailing spaces, or a semicolon (;), or begin with either a single quote (') or a double quote (").

Once you have supplied all the creation parameters by setting the appropriate properties on a ULCreateParms object, you create a creation parameters string using the ULCreateParms.ToString. The resulting string can then be used as the createParms parameter of the ULDatabaseManager.CreateDatabase(string,byte[],string) method.

For more information, see [“UltraLite Connection String Parameters Reference” \[UltraLite - Database Management and Reference\]](#).

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows CE device. The database is created case sensitive and with the UTF8 character set.

```
' Visual Basic
Dim createParms As ULCreateParms = New ULCreateParms
createParms.CaseSensitive = True
createParms.UTF8Encoding = True
Dim openParms As ULConnectionParms = New ULConnectionParms
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"
' Assumes file coll_1250LATIN2.vb is
' also compiled in the current project
ULConnection.DatabaseManager.CreateDatabase( _
    openParms.ToString(), _
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data, _
    createParms.ToString() _
)
Dim conn As ULConnection = _
    New ULConnection( openParms.ToString() )
conn.Open()

// C#
ULCreateParms createParms = new ULCreateParms();
createParms.CaseSensitive = true;
createParms.UTF8Encoding = true;
```

```
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnCE = @"\"UltraLite\\MyDatabase.udb";
// Assumes file coll_1250LATIN2.vb is
// also compiled in the current project
ULConnection.DatabaseManager.CreateDatabase(
    openParms.ToString(),
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data,
    createParms.ToString()
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

See also

- ◆ [“ULCreateParms members” on page 608](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)
- ◆ [“CreateDatabase method” on page 641](#)
- ◆ [“ToString method” on page 617](#)

ULCreateParms members

Public constructors

Member name	Description
ULCreateParms constructor	Initializes a ULCreateParms instance with its default values.

Public properties

Member name	Description
CaseSensitive property	Specifies whether the new database should be case sensitive when comparing string values.
ChecksumLevel property	Specifies the level of database page checksums enabled for the new database.
DateFormat property	Specifies the date format used for string conversions by the new database.
DateOrder property	Specifies the date order used for string conversions by the new database.
FIPS property	Specifies whether the new database should be using AES_FIPS encryption or AES encryption.
MaxHashSize property	Specifies the default maximum number of bytes to use for index hashing in the new database.
NearestCentury property	Specifies the nearest century used for string conversions by the new database.
Obfuscate property	Specifies whether the new database should be using obfuscation (simple encryption) or not.

Member name	Description
PageSize property	Specifies the page size of the new database, in bytes or kilobytes.
Precision property	Specifies the floating-point precision used for string conversions by the new database.
Scale property	Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the new database.
TimeFormat property	Specifies the time format used for string conversions by the new database.
TimestampFormat property	Specifies the timestamp format used for string conversions by the new database.
TimestampIncrement property	Specifies the minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).
UTF8Encoding property	Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Public methods

Member name	Description
ToString method	Returns the string representation of this instance.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)
- ◆ [“CreateDatabase method” on page 641](#)
- ◆ [“ToString method” on page 617](#)

ULCreateParms constructor

Initializes a ULCreateParms instance with its default values.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULCreateParms();
```

See also

- ◆ [“ULCreateParms class” on page 607](#)

- ◆ [“ULCreateParms members” on page 608](#)

CaseSensitive property

Specifies whether the new database should be case sensitive when comparing string values.

Syntax

Visual Basic

Public Property **CaseSensitive** As Boolean

C#

```
public bool CaseSensitive { get; set; }
```

Property value

True if the database should be case sensitive, false if the database should be case insensitive. The default is false.

Remarks

CaseSensitive only affects how string data is compared and sorted. Database identifiers such as table names, column names, index names, and connection user IDs are always case insensitive. Connection passwords and database encryption keys are always case sensitive.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

ChecksumLevel property

Specifies the level of database page checksums enabled for the new database.

Syntax

Visual Basic

Public Property **ChecksumLevel** As Integer

C#

```
public int ChecksumLevel { get; set; }
```

Property value

An integer specifying the checksum level. Valid values are 0, 1, and 2. The default is 0.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

DateFormat property

Specifies the date format used for string conversions by the new database.

Syntax

Visual Basic

Public Property **DateFormat** As String

C#

public string **DateFormat** { get; set; }

Property value

A string specifying the date format. If the value is a null reference (Nothing in Visual Basic), the database will use "YYYY-MM-DD". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

DateOrder property

Specifies the date order used for string conversions by the new database.

Syntax

Visual Basic

Public Property **DateOrder** As UDateOrder

C#

public UDateOrder **DateOrder** { get; set; }

Property value

A UDateOrder value identifying the date order for string conversions. The default is YMD.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)
- ◆ [“UDateOrder enumeration” on page 694](#)

FIPS property

Specifies whether the new database should be using AES_FIPS encryption or AES encryption.

Syntax**Visual Basic**

Public Property **FIPS** As Boolean

C#

```
public bool FIPS { get; set; }
```

Property value

True if the database should be encrypted using AES_FIPS, false if the database should be encrypted with AES. The default is false.

Remarks

Encryption must be turned on by supplying a value for the connection parameter EncryptionKey when the new database is created. If FIPS is set true and no encryption key is supplied, the ULDatabaseManager.CreateDatabase(string,byte[],string) method will fail with a missing encryption key error.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)
- ◆ [“EncryptionKey property” on page 586](#)
- ◆ [“CreateDatabase method” on page 641](#)

MaxHashSize property

Specifies the default maximum number of bytes to use for index hashing in the new database.

Syntax**Visual Basic**

Public Property **MaxHashSize** As Integer

C#

```
public int MaxHashSize { get; set; }
```

Property value

An integer specifying the maximum hash size. The value must be in the range [0,32]. The default is 8.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

NearestCentury property

Specifies the nearest century used for string conversions by the new database.

Syntax

Visual Basic

Public Property **NearestCentury** As Integer

C#

```
public int NearestCentury { get; set; }
```

Property value

An integer specifying the nearest century. The value must be in the range [0,100]. The default is 50.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

Obfuscate property

Specifies whether the new database should be using obfuscation (simple encryption) or not.

Syntax

Visual Basic

Public Property **Obfuscate** As Boolean

C#

```
public bool Obfuscate { get; set; }
```

Property value

True if the database should be encrypted using obfuscation, false if the database should not be obfuscated. The default is false.

Remarks

This option is ignored if FIPS encryption is turned on (ULCreateParms.FIPS). If obfuscation is turned on and a value is supplied for the connection parameter EncryptionKey (DBKEY) when the new database is created, the encryption key will be ignored.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)
- ◆ [“FIPS property” on page 611](#)

PageSize property

Specifies the page size of the new database, in bytes or kilobytes.

Syntax

Visual Basic

Public Property **PageSize** As Integer

C#

```
public int PageSize { get; set; }
```

Property value

An integer specifying the page size in bytes. Valid values are 1024 (1K), 2048 (2K), 4096 (4K), 8192 (8K), 16384 (16K). The default is 4096.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

Precision property

Specifies the floating-point precision used for string conversions by the new database.

Syntax

Visual Basic

Public Property **Precision** As Integer

C#

```
public int Precision { get; set; }
```

Property value

An integer specifying the precision. The value must be in the range [1,127]. The default is 30.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)
- ◆ [“Scale property” on page 614](#)

Scale property

Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the new database.

Syntax**Visual Basic**

Public Property **Scale** As Integer

C#

```
public int Scale { get; set; }
```

Property value

An integer specifying the scale. The value must be in the range [0,127]. The default is 6.

Remarks

Scale must be less than or equal to the Precision. If Scale is greater than the Precision, an error will occur while creating the database.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

TimeFormat property

Specifies the time format used for string conversions by the new database.

Syntax**Visual Basic**

Public Property **TimeFormat** As String

C#

```
public string TimeFormat { get; set; }
```

Property value

A string specifying the time format. If the value is a null reference (Nothing in Visual Basic), the database will use "HH:NN:SS.SSS". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

TimestampFormat property

Specifies the timestamp format used for string conversions by the new database.

Syntax

Visual Basic

Public Property **TimestampFormat** As String

C#

public string **TimestampFormat** { get; set; }

Property value

A string specifying the timestamp format. If the value is a null reference (Nothing in Visual Basic), the database will use "YYYY-MM-DD HH:NN:SS.SSS". In C#, you must escape any backslash characters in paths or use @-quoted string literals. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

TimestampIncrement property

Specifies the minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).

Syntax

Visual Basic

Public Property **TimestampIncrement** As Integer

C#

public int **TimestampIncrement** { get; set; }

Property value

An integer specifying the timestamp increment. The value must be in the range [1,60000000]. The default is 1.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

UTF8Encoding property

Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Syntax

Visual Basic

Public Property **UTF8Encoding** As Boolean

C#

```
public bool UTF8Encoding { get; set; }
```

Property value

True if the database should use the UTF8 character set, false if the database should use the character set associated with the collation. The default is false.

Remarks

Choose to use the UTF8 character set if you wish to store characters that are not in the character set associated with the collation. For example, you create a database with the 1252LATIN1 collation because you want US sorting but specify UTF8Encoding true because you want to store international addresses as they are spelled locally.

For databases used on Symbian OS devices, you must set UTF8Encoding true.

For databases used on Palm OS devices, you must leave UTF8Encoding false.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

ToString method

Returns the string representation of this instance.

Syntax

Visual Basic

Public Overrides Function **ToString()** As String

C#

```
public override string ToString();
```

Return value

The string representation of this instance as a semicolon-separated list keyword=value pairs.

See also

- ◆ [“ULCreateParms class” on page 607](#)
- ◆ [“ULCreateParms members” on page 608](#)

ULCursorSchema class

UL Ext.: Represents the schema of an UltraLite.NET cursor. This class is abstract and so cannot be instantiated.

Syntax

Visual Basic

MustInherit Public Class **ULCursorSchema**

C#

public abstract class **ULCursorSchema**

Remarks

This class is an abstract base class of the ULTableSchema class and the ULResultSetSchema class.

Note to users porting from the iAnywhere.UltraLite namespace: Column IDs are 0-based, not 1-based as they are in the iAnywhere.UltraLite namespace.

See also

- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULResultSetSchema class” on page 816](#)

ULCursorSchema members

Public properties

Member name	Description
ColumnCount property	Returns the number of columns in the cursor.
IsOpen property	Checks whether the cursor schema is currently open.
Name property	Returns the name of the cursor.

Public methods

Member name	Description
GetColumnID method	Returns the column ID of the named column.
GetColumnName method	Returns the name of the column identified by the specified column ID.
GetColumnPrecision method	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Member name	Description
GetColumnSQLName method	Returns the name of the column identified by the specified column ID.
GetColumnScale method	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSize method	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnULDbType method	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetSchemaTable method	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULResultSetSchema class” on page 816](#)

ColumnCount property

Returns the number of columns in the cursor.

Syntax

Visual Basic

Public Readonly Property **ColumnCount** As Short

C#

```
public short ColumnCount { get;}
```

Property value

The number of columns in the cursor or 0 if the cursor schema is closed.

Remarks

Column IDs range from 0 to ColumnCount-1, inclusive.

Column IDs and count might change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)

IsOpen property

Checks whether the cursor schema is currently open.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen { get;}
```

Property value

True if the cursor schema is currently open, false if the cursor schema is closed.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)

Name property

Returns the name of the cursor.

Syntax

Visual Basic

Public Readonly Property **Name** As String

C#

```
public string Name { get;}
```

Property value

The name of the cursor as a string.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)

GetColumnID method

Returns the column ID of the named column.

Syntax

Visual Basic

```
Public Function GetColumnID( _  
    ByVal name As String _  
) As Short
```

C#

```
public short GetColumnID(  
    string name  
);
```

Parameters

- ◆ **name** The name of the column.

Return value

The column ID of the named column.

Remarks

Column IDs range from 0 to ColumnCount-1, inclusive.

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and counts might change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnName method

Returns the name of the column identified by the specified column ID.

Syntax

Visual Basic

```
Public Function GetColumnName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and count may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnPrecision method

Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Syntax**Visual Basic**

```
Public Function GetColumnPrecision( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnPrecision(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The precision of the specified numeric column.

See also

- ◆ [“ULCursorSchema class” on page 618](#)

- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnSQLName method

Returns the name of the column identified by the specified column ID.

Syntax

Visual Basic

```
Public Function GetColumnSQLName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnSQLName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are using aliases, the name of the column is the alias.

The GetColumnSQLName method differs from the GetColumnName in that for non-aliased, non-computed columns GetColumnSQLName always returns just the name of the column (without the table name as a prefix). While this behavior more closely resembles the behavior of other ADO.NET providers, it is more likely to produce non-unique names.

Column IDs and count may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“GetColumnName method” on page 621](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnScale method

Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).

Syntax

Visual Basic

```
Public Function GetColumnScale( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnScale(  
    int columnID  
);
```

Parameters

- ♦ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The scale of the specified numeric column.

See also

- ♦ [“ULCursorSchema class” on page 618](#)
- ♦ [“ULCursorSchema members” on page 618](#)
- ♦ [“GetColumnULDbType method” on page 625](#)
- ♦ [“ColumnCount property” on page 619](#)

GetColumnSize method

Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).

Syntax

Visual Basic

```
Public Function GetColumnSize( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public int GetColumnSize(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

The size of the specified sized column.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnULDbType method

Returns the UltraLite.NET data type of the column identified by the specified column ID.

Syntax**Visual Basic**

```
Public Function GetColumnULDbType( _  
    ByVal columnID As Integer _  
) As UDbType
```

C#

```
public UDbType GetColumnULDbType(  
    int columnID  
);
```

Parameters

- ◆ **columnID** ID of the column. The value must be in the range [0,ColumnCount-1].

Return value

A UDbType enumerated integer.

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ULDbType enumeration” on page 695](#)

GetSchemaTable method

Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

Syntax

Visual Basic

Public Function **GetSchemaTable()** As DataTable

C#

public DataTable **GetSchemaTable();**

Return value

A System.Data.DataTable that describes the column schema.

Remarks

For more information, see [“GetSchemaTable method” on page 681](#).

See also

- ◆ [“ULCursorSchema class” on page 618](#)
- ◆ [“ULCursorSchema members” on page 618](#)
- ◆ [DataTable](#)
- ◆ [“ULDataReader class” on page 656](#)

ULDataAdapter class

Represents a set of commands and a database connection used to fill a System.Data.DataSet and to update a database. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULDataAdapter**
Inherits DbDataAdapter

C#

public sealed class **ULDataAdapter** : DbDataAdapter

Remarks

The System.Data.DataSet provides a way to work with data offline; that is, away from your UltraLite database. The ULDataAdapter provides methods to associate a System.Data.DataSet with a set of SQL statements.

Since UltraLite is a local database and MobiLink has conflict resolution, the use of the ULDataAdapter is limited. For most purposes, the ULDataReader or the ULTable provide more efficient access to data.

Inherits: System.Data.Common.DbDataAdapter

Implements: System.Data.IDbDataAdapter, System.Data.IDataAdapter, System.IDisposable

See also

- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [DbDataAdapter](#)
- ◆ [IDbDataAdapter](#)
- ◆ [IDataAdapter](#)
- ◆ [IDisposable](#)

ULDataAdapter members

Public constructors

Member name	Description
ULDataAdapter constructors	Initializes a new instance of the “ULDataAdapter class” on page 627 .

Public properties

Member name	Description
AcceptChangesDuringFill (inherited from <code>DataAdapter</code>)	
AcceptChangesDuringUpdate (inherited from <code>DataAdapter</code>)	
ContinueUpdateOnError (inherited from <code>DataAdapter</code>)	
DeleteCommand property	Specifies a <code>ULCommand</code> object that is executed against the database when <code>DbDataAdapter.Update(System.Data.DataSet)</code> is called to delete rows in the database that correspond to deleted rows in the <code>System.Data.DataSet</code> .
FillLoadOption (inherited from <code>DataAdapter</code>)	
InsertCommand property	Specifies a <code>ULCommand</code> object that is executed against the database when <code>DbDataAdapter.Update(System.Data.DataSet)</code> is called to insert rows in the database that correspond to inserted rows in the <code>System.Data.DataSet</code> .
MissingMappingAction (inherited from <code>DataAdapter</code>)	
MissingSchemaAction (inherited from <code>DataAdapter</code>)	
ReturnProviderSpecificTypes (inherited from <code>DataAdapter</code>)	
SelectCommand property	Specifies a <code>ULCommand</code> that is used during <code>System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet)</code> or <code>System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataSet, System.Data.SchemaType)</code> to obtain a result set from the database for copying into a <code>System.Data.DataSet</code> .
TableMappings property	Returns a collection that provides the master mapping between a source table and a <code>System.Data.DataTable</code>
UpdateBatchSize (inherited from <code>DbDataAdapter</code>)	
UpdateCommand property	Specifies a <code>ULCommand</code> object that is executed against the database when <code>System.Data.Common.DbDataAdapter.Update(System.Data.DataSet)</code> is called to update rows in the database that correspond to updated rows in the <code>System.Data.DataSet</code> .

Public methods

Member name	Description
Fill (inherited from DbDataAdapter)	Fills a DataSet or a DataTable .
FillSchema (inherited from DbDataAdapter)	Adds a DataTable to a DataSet and configures the schema to match that in the data source.
GetFillParameters method	Returns the parameters set by the user when executing a SELECT statement.
ResetFillLoadOption (inherited from DataAdapter)	
ShouldSerializeAcceptChangesDuringFill (inherited from DataAdapter)	
ShouldSerializeFillLoadOption (inherited from DataAdapter)	
Update (inherited from DbDataAdapter)	Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the DataSet .

Public events

Member name	Description
FillError (inherited from DataAdapter)	
RowUpdated event	Occurs during an update after a command is executed against the data source. When an attempt to update is made, the event fires.
RowUpdating event	Occurs during an update before a command is executed against the data source. When an attempt to update is made, the event fires.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [DataSet](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULTable class” on page 904](#)
- ◆ [DbDataAdapter](#)
- ◆ [IDbDataAdapter](#)
- ◆ [IDataAdapter](#)
- ◆ [IDisposable](#)

ULDataAdapter constructors

Initializes a new instance of the [“ULDataAdapter class” on page 627](#).

ULDataAdapter() constructor

Initializes a ULDataAdapter object.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULDataAdapter();
```

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULDataAdapter constructors” on page 630](#)
- ◆ [“ULDataAdapter\(ULCommand\) constructor” on page 630](#)
- ◆ [“ULDataAdapter\(String, ULConnection\) constructor” on page 631](#)
- ◆ [“ULDataAdapter\(String, String\) constructor” on page 632](#)

ULDataAdapter(ULCommand) constructor

Initializes a ULDataAdapter object with the specified SELECT statement.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal selectCommand As ULCommand _  
)
```

C#

```
public ULDataAdapter(  
    ULCommand selectCommand  
);
```

Parameters

- ◆ **selectCommand** A ULCommand object that is used during System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet) to select records from the data source for placement in the System.Data.DataSet.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULDataAdapter constructors” on page 630](#)
- ◆ [“ULDataAdapter\(\) constructor” on page 630](#)
- ◆ [“ULDataAdapter\(String, ULConnection\) constructor” on page 631](#)
- ◆ [“ULDataAdapter\(String, String\) constructor” on page 632](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbDataAdapter.Fill](#)
- ◆ [DataSet](#)

ULDataAdapter(String, ULConnection) constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal selectCommandText As String, _  
    ByVal selectConnection As ULConnection _  
)
```

C#

```
public ULDataAdapter(  
    string selectCommandText,  
    ULConnection selectConnection  
);
```

Parameters

- ◆ **selectCommandText** A SELECT statement to be used by the ULDataAdapter.SelectCommand of the ULDataAdapter.
- ◆ **selectConnection** A ULConnection object that defines a connection to a database.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULDataAdapter constructors” on page 630](#)
- ◆ [“ULDataAdapter\(\) constructor” on page 630](#)
- ◆ [“ULDataAdapter\(ULCommand\) constructor” on page 630](#)
- ◆ [“ULDataAdapter\(String, String\) constructor” on page 632](#)
- ◆ [“SelectCommand property” on page 634](#)
- ◆ [“ULConnection class” on page 542](#)

ULDataAdapter(String, String) constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal selectCommandText As String, _  
    ByVal selectConnectionString As String _  
)
```

C#

```
public ULDataAdapter(  
    string selectCommandText,  
    string selectConnectionString  
);
```

Parameters

- ♦ **selectCommandText** A SELECT statement to be used by the ULDataAdapter.SelectCommand of the ULDataAdapter.
- ♦ **selectConnectionString** A connection string for an UltraLite.NET database.

See also

- ♦ [“ULDataAdapter class” on page 627](#)
- ♦ [“ULDataAdapter members” on page 627](#)
- ♦ [“ULDataAdapter constructors” on page 630](#)
- ♦ [“ULDataAdapter\(\) constructor” on page 630](#)
- ♦ [“ULDataAdapter\(ULCommand\) constructor” on page 630](#)
- ♦ [“ULDataAdapter\(String, ULConnection\) constructor” on page 631](#)
- ♦ [“SelectCommand property” on page 634](#)

DeleteCommand property

Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update (System.Data.DataSet) is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

Syntax

Visual Basic

```
Public Property DeleteCommand As ULCommand
```

C#

```
public ULCommand DeleteCommand { get; set; }
```

Property value

A ULCommand object that is executed to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

Remarks

When DeleteCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The DeleteCommand maintains a reference to the existing ULCommand.

This is the strongly-typed version of System.Data.IDbDataAdapter.DeleteCommand and System.Data.Common.DbDataAdapter.DeleteCommand.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DataSet](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [IDbDataAdapter.DeleteCommand](#)
- ◆ [DbDataAdapter.DeleteCommand](#)

InsertCommand property

Specifies a ULCommand object that is executed against the database when DbDataAdapter.Update (System.Data.DataSet) is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

Syntax**Visual Basic**

```
Public Property InsertCommand As ULCommand
```

C#

```
public ULCommand InsertCommand { get; set; }
```

Property value

A ULCommand object that is executed to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

Remarks

When InsertCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The InsertCommand maintains a reference to the existing ULCommand.

This is the strongly-typed version of System.Data.IDbDataAdapter.InsertCommand and System.Data.Common.DbDataAdapter.InsertCommand.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [DataSet](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [IDbDataAdapter.InsertCommand](#)
- ◆ [DbDataAdapter.InsertCommand](#)

SelectCommand property

Specifies a ULCommand that is used during System.Data.Common.DbDataAdapter.Fill (System.Data.DataSet) or System.Data.Common.DbDataAdapter.FillSchema (System.Data.DataSet, System.Data.SchemaType) to obtain a result set from the database for copying into a System.Data.DataSet.

Syntax**Visual Basic**

Public Property **SelectCommand** As ULCommand

C#

```
public ULCommand SelectCommand { get; set; }
```

Property value

A ULCommand object that is executed to fill the System.Data.DataSet.

Remarks

When SelectCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The SelectCommand maintains a reference to the existing ULCommand.

If the SelectCommand does not return any rows, no tables are added to the System.Data.DataSet, and no exception is raised. The SELECT statement can also be specified in the ULDataAdapter(ULCommand)"/>, ULDataAdapter(String,ULConnection), or ULDataAdapter(String,String) constructors.

This is the strongly-typed version of System.Data.IDbDataAdapter.SelectCommand and System.Data.Common.DbDataAdapter.SelectCommand.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DataSet](#)
- ◆ [DbDataAdapter.Fill](#)
- ◆ [DbDataAdapter.FillSchema](#)
- ◆ [“ULDataAdapter\(ULCommand\) constructor” on page 630](#)

- ◆ [“ULDataAdapter\(String, ULConnection\) constructor” on page 631](#)
- ◆ [“ULDataAdapter\(String, String\) constructor” on page 632](#)
- ◆ [IDbDataAdapter.SelectCommand](#)
- ◆ [DbDataAdapter.SelectCommand](#)

TableMappings property

Returns a collection that provides the master mapping between a source table and a System.Data.DataTable

Syntax

Visual Basic

Public Readonly Property **TableMappings** As DataTableMappingCollection

C#

public DataTableMappingCollection **TableMappings** { get; }

Property value

A collection of System.Data.Common.DataTableMapping objects providing the master mapping between source tables and System.Data.DataTables. The default value is an empty collection.

Remarks

When reconciling changes, the ULDataAdapter uses the System.Data.Common.DataTableMappingCollection collection to associate the column names used by the data source with the column names used by the System.Data.DataSet.

This is the strongly-typed version of System.Data.IDataAdapter.TableMappings.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [DataTable](#)
- ◆ [DataTableMapping](#)
- ◆ [DataTable](#)
- ◆ [DataTableMappingCollection](#)
- ◆ [DataSet](#)
- ◆ [IDataAdapter.TableMappings](#)

UpdateCommand property

Specifies a ULCommand object that is executed against the database when System.Data.Common.DbDataAdapter.Update(System.Data.DataSet) is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Syntax**Visual Basic**

Public Property **UpdateCommand** As ULCommand

C#

public ULCommand **UpdateCommand** { get; set; }

Property value

A ULCommand object that is executed to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Remarks

When UpdateCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The UpdateCommand maintains a reference to the existing ULCommand.

If execution of this command returns rows, these rows may be merged with the System.Data.DataSet depending on how you set the ULCommand.UpdatedRowSource of the ULCommand object.

This is the strongly-typed version of System.Data.IDbDataAdapter.UpdateCommand and System.Data.Common.DbDataAdapter.DeleteCommand.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DataSet](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [“UpdatedRowSource property” on page 509](#)
- ◆ [IDbDataAdapter.UpdateCommand](#)
- ◆ [DbDataAdapter.DeleteCommand](#)

GetFillParameters method

Returns the parameters set by the user when executing a SELECT statement.

Syntax**Visual Basic**

Public Function **GetFillParameters()** As ULParameter

C#

public ULParameter **GetFillParameters();**

Return value

An array of ULParameter objects that contains the parameters set by the user.

Remarks

This is the strongly-typed version of `System.Data.Common.DbDataAdapter.GetFillParameters`.

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [DbDataAdapter.GetFillParameters](#)
- ◆ [“ULParameter class” on page 751](#)

RowUpdated event

Occurs during an update after a command is executed against the data source. When an attempt to update is made, the event fires.

Syntax**Visual Basic**

Public Event **RowUpdated** As ULRowUpdatedEventHandler

C#

public event ULRowUpdatedEventHandler **RowUpdated** ;

Remarks

To process row updated events, you must create a `ULRowUpdatedEventHandler` delegate and attach it to this event.

Event data

- ◆ **Command** Returns the `ULCommand` executed when `DbDataAdapter.Update(System.Data.DataRow [], System.Data.Common.DataTableMapping)` is called.
- ◆ **RecordsAffected** Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement. For SELECT statements this value is -1.
- ◆ **Command**
- ◆ **Errors**
- ◆ **Row**
- ◆ **RowCount**
- ◆ **StatementType**
- ◆ **Status**
- ◆ **TableMapping**

See also

- ◆ [“ULDataAdapter class” on page 627](#)

- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULRowUpdatedEventHandler delegate” on page 827](#)

RowUpdating event

Occurs during an update before a command is executed against the data source. When an attempt to update is made, the event fires.

Syntax

Visual Basic

Public Event **RowUpdating** As ULRowUpdatingEventHandler

C#

public event ULRowUpdatingEventHandler **RowUpdating** ;

Remarks

To process row updating events, you must create a ULRowUpdatingEventHandler delegate and attach it to this event.

Event data

- ◆ **Command** Specifies the ULCommand to execute when performing the DbDataAdapter.Update (System.Data.DataRow[], System.Data.Common.DataTableMapping).
- ◆ **Command**
- ◆ **Errors**
- ◆ **Row**
- ◆ **StatementType**
- ◆ **Status**
- ◆ **TableMapping**

See also

- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataAdapter members” on page 627](#)
- ◆ [“ULRowUpdatedEventHandler delegate” on page 827](#)

ULDatabaseManager class

UL Ext.: Manages synchronization listeners and the UltraLite.NET runtime type. The ULDatabaseManager class also allows you to drop (delete) UltraLite.NET databases. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULDatabaseManager**

C#

public sealed class **ULDatabaseManager**

Remarks

This class is a singleton class whose only instance is accessible through the static (Shared in Visual Basic) `ULConnection.DatabaseManager`.

To use the UltraLite Engine runtime of UltraLite.NET, set `ULDatabaseManager.RuntimeType` to the appropriate value before using any other UltraLite.NET API.

Example

The following example selects the UltraLite Engine runtime and creates a connection.

```
' Visual Basic
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT
Dim conn As ULConnection = new ULConnection
' The RuntimeType is now locked

// C#
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT;
ULConnection conn = new ULConnection();
// The RuntimeType is now locked
```

See also

- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“DatabaseManager property” on page 550](#)
- ◆ [“RuntimeType property” on page 640](#)

ULDatabaseManager members

Public properties

Member name	Description
RuntimeType property	Specifies the UltraLite.NET runtime type. The runtime type must be selected before using any other UltraLite.NET API.

Public methods

Member name	Description
CreateDatabase method	Creates a new UltraLite database.
DropDatabase method	Deletes the specified database. You cannot drop a database that has open connections.
SetActiveSyncListener method	Specifies the listener object used to process ActiveSync calls from the MobiLink provider for ActiveSync.
SetServerSyncListener method	Specifies the listener object used to process the specified server synchronization message.
SignalSyncIsComplete method	Signals the MobiLink provider for ActiveSync that an application has completed synchronization.

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“DatabaseManager property” on page 550](#)
- ◆ [“RuntimeType property” on page 640](#)

RuntimeType property

Specifies the UltraLite.NET runtime type. The runtime type must be selected before using any other UltraLite.NET API.

Syntax**Visual Basic**

Public Shared Property **RuntimeType** As ULRuntimeType

C#

```
public const ULRuntimeType RuntimeType { get; set; }
```

Property value

A ULRuntimeType value identifying the type of the unmanaged UltraLite.NET runtime.

Example

The following example selects the UltraLite Engine runtime and creates a connection.

```
' Visual Basic
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT
Dim conn As ULConnection = new ULConnection
' The RuntimeType is now locked

// C#
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT;
```

```
ULConnection conn = new ULConnection();  
// The RuntimeType is now locked
```

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“ULRuntimeType enumeration” on page 832](#)

CreateDatabase method

Creates a new UltraLite database.

Syntax**Visual Basic**

```
Public Sub CreateDatabase( _  
    ByVal connString As String, _  
    ByVal collationData As Byte(), _  
    ByVal createParms As String _  
)
```

C#

```
public void CreateDatabase(  
    string connString,  
    byte[] collationData,  
    string createParms  
);
```

Parameters

- ◆ **connString** The parameters for identifying a database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [“ULConnectionParms class” on page 578](#).
- ◆ **collationData** The collation data specifying how the database will store and compare strings.
- ◆ **createParms** The parameters used to configure the new database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [“ULCreateParms class” on page 607](#).

Remarks

To specify a collation, you must first include the appropriate collation data source file from the src\ulcollations\cs (for C# projects) or src\ulcollations\vb.net (for Visual Basic projects) subdirectory of your SQL Anywhere installation directory. Once included in your project, use the collation's Data property to supply the collation data to the CreateDatabase() method.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows CE device then opens a connection to it.

```
' Visual Basic  
Dim openParms As ULConnectionParms = New ULConnectionParms
```

```
openParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"
' Assumes file src\ulcollations\vb.net\coll_1250LATIN2.vb is
' also compiled in the current project
ULConnection.DatabaseManager.CreateDatabase( _
    openParms.ToString(), _
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data, _
    "" _
)
Dim conn As ULConnection = _
    New ULConnection( openParms.ToString() )
conn.Open()

// C#
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnCE = @"\\UltraLite\\MyDatabase.udb";
// Assumes file src\ulcollations\cs\coll_1250LATIN2.cs is
// also compiled in the current project
ULConnection.DatabaseManager.CreateDatabase(
    openParms.ToString(),
    iAnywhere.UltraLite.Collations.Collation_1250LATIN2.Data,
    ""
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“Open method” on page 570](#)

DropDatabase method

Deletes the specified database.

You cannot drop a database that has open connections.

Syntax

Visual Basic

```
Public Sub DropDatabase( _
    ByVal connString As String _
)
```

C#

```
public void DropDatabase(
    string connString
);
```

Parameters

- ◆ **connString** The parameters for identifying a database in the form of a semicolon-separated list of keyword-value pairs. For more information, see the [“ULConnectionParms class” on page 578](#).

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows CE device then opens a connection to it.

```
' Visual Basic
Dim connParms As ULConnectionParms = New ULConnectionParms
connParms.DatabaseOnCE = "\UltraLite\MyDatabase.udb"
ULConnection.DatabaseManager.DropDatabase( _
    connParms.ToString() _
)

// C#
ULConnectionParms connParms = new ULConnectionParms();
connParms.DatabaseOnCE = @"\UltraLite\MyDatabase.udb";
ULConnection.DatabaseManager.DropDatabase(
    connParms.ToString()
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“Open method” on page 570](#)

SetActiveSyncListener method

Specifies the listener object used to process ActiveSync calls from the MobiLink provider for ActiveSync.

Syntax

Visual Basic

```
Public Sub SetActiveSyncListener( _
    ByVal appClassName As String, _
    ByVal listener As ULActiveSyncListener _
)
```

C#

```
public void SetActiveSyncListener(
    string appClassName,
    ULActiveSyncListener listener
);
```

Parameters

- ◆ **appClassName** The unique class name for the application. This is the class name used when the application is registered for use with ActiveSync.
- ◆ **listener** The ULActiveSyncListener object. Use null (Nothing in Visual Basic) to remove the previous listener.

Remarks

The parameter *appClassName* is the unique identifier used to identify the application. The application can only use one *appClassName* at a time. While a listener is registered with a particular *appClassName*, calls to *SetServerSyncListener* or *SetActiveSyncListener* with a different *appClassName* fail.

To remove the *ActiveSync* listener, call *SetActiveSyncListener* with a null reference (Nothing in Visual Basic) as the *listener* parameter.

To remove all listeners, call *SetServerSyncListener* with a null reference (Nothing in Visual Basic) for all parameters.

Applications should remove all listeners prior to exiting.

Example

For an example of *SetActiveSyncListener*, see the [“ActiveSyncInvoked method” on page 463](#).

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“SetServerSyncListener method” on page 644](#)
- ◆ [“SetActiveSyncListener method” on page 643](#)
- ◆ [“ULActiveSyncListener interface” on page 463](#)
- ◆ [“ActiveSyncInvoked method” on page 463](#)

SetServerSyncListener method

Specifies the listener object used to process the specified server synchronization message.

Syntax

Visual Basic

```
Public Sub SetServerSyncListener( _  
    ByVal messageName As String, _  
    ByVal appClassName As String, _  
    ByVal listener As ULServerSyncListener _  
)
```

C#

```
public void SetServerSyncListener(  
    string messageName,  
    string appClassName,  
    ULServerSyncListener listener  
);
```

Parameters

- ◆ **messageName** The name of the message.
- ◆ **appClassName** The unique class name for the application. This is a unique identifier used to identify the application.

- ◆ **listener** The ULServerSyncListener object. Use null (Nothing in Visual Basic) to remove the previous listener.

Remarks

The parameter *appClassName* is the unique identifier used to identify the application. The application may only use one *appClassName* at a time. While a listener is registered with a particular *appClassName*, calls to SetServerSyncListener or SetActiveSyncListener with a different *appClassName* fail.

To remove the listener for a particular message, call SetServerSyncListener with a null reference (Nothing in Visual Basic) as the *listener* parameter.

To remove all listeners, call SetServerSyncListener with a null reference (Nothing in Visual Basic) for all parameters.

Applications should remove all listeners before exiting.

Example

For an example of SetServerSyncListener, see the [“ServerSyncInvoked method” on page 833](#).

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“SetServerSyncListener method” on page 644](#)
- ◆ [“SetActiveSyncListener method” on page 643](#)
- ◆ [“ServerSyncInvoked method” on page 833](#)

SignalSyncIsComplete method

Signals the MobiLink provider for ActiveSync that an application has completed synchronization.

Syntax

Visual Basic

```
Public Sub SignalSyncIsComplete()
```

C#

```
public void SignalSyncIsComplete();
```

Example

For an example of SignalSyncIsComplete, see the [“ActiveSyncInvoked method” on page 463](#).

See also

- ◆ [“ULDatabaseManager class” on page 639](#)
- ◆ [“ULDatabaseManager members” on page 639](#)
- ◆ [“SignalSyncIsComplete method” on page 645](#)

ULDatabaseSchema class

UL Ext.: Represents the schema of an UltraLite.NET database. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULDatabaseSchema**

C#

public sealed class **ULDatabaseSchema**

Remarks

There is no constructor for this class. A ULDatabaseSchema object is attached to a connection as its ULConnection.Schema and is only valid while that connection is open.

See also

- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“Schema property” on page 552](#)

ULDatabaseSchema members

Public properties

Member name	Description
CollationName property	The name of the database's collation sequence.
IsCaseSensitive property	Checks whether the database is case sensitive.
IsOpen property	Whether the database schema is open.
PublicationCount property	The number of publications in the database.
TableCount property	The number of tables in the database.

Public methods

Member name	Description
GetDatabaseProperty method	Returns the value of the specified database property.
GetPublicationName method	Returns the name of the publication identified by the specified publication ID. Publication IDs are not publication masks.
GetPublicationSchema method	Returns the publication schema corresponding to the named publication.

Member name	Description
GetTableCountInPublications method	Returns the number of tables included in the specified publication mask.
GetTableName method	Returns the name of the table identified by the specified table ID.
SetDatabaseOption method	Sets the value for the specified database option.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“Schema property” on page 552](#)

CollationName property

The name of the database's collation sequence.

Syntax**Visual Basic**

Public Readonly Property **CollationName** As String

C#

```
public string CollationName { get; }
```

Property value

A string representing the database's collation sequence.

Remarks

The database collation sequence affects how indexes on tables and result sets are sorted.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)

IsCaseSensitive property

Checks whether the database is case sensitive.

Syntax**Visual Basic**

Public Readonly Property **IsCaseSensitive** As Boolean

C#

```
public bool IsCaseSensitive { get;}
```

Property value

True if the database is case sensitive, and false if the database is case insensitive.

Remarks

Database case sensitivity affects how indexes on tables and result sets are sorted. Case sensitivity also affects how `ULConnectionParms.UserID` and `ULConnectionParms.Password` are verified.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)
- ◆ [“UserID property” on page 587](#)
- ◆ [“Password property” on page 586](#)

IsOpen property

Whether the database schema is open.

Syntax**Visual Basic**

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen { get;}
```

Property value

True if this database schema is currently open, false if this database schema is currently closed.

Remarks

A `ULDatabaseSchema` object is open only if the connection it is attached to is open.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)

PublicationCount property

The number of publications in the database.

Syntax**Visual Basic**

Public Readonly Property **PublicationCount** As Integer

C#

```
public int PublicationCount { get;}
```

Property value

The number of publications in the database or zero if the connection is not open.

Remarks

Publication IDs range from 1 to PublicationCount, inclusively. Publication IDs are not publication masks.

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“GetPublicationName method” on page 652](#)

TableCount property

The number of tables in the database.

Syntax**Visual Basic**

Public Readonly Property **TableCount** As Integer

C#

```
public int TableCount { get;}
```

Property value

The number of tables in the database or zero if the connection is not open.

Remarks

Table IDs range from 1 to TableCount, inclusively.

Note: Table IDs and counts may change during a schema upgrade. To correctly identify a table, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)

GetDatabaseProperty method

Returns the value of the specified database property.

Syntax

Visual Basic

```
Public Function GetDatabaseProperty( _  
    ByVal name As String _  
) As String
```

C#

```
public string GetDatabaseProperty(  
    string name  
);
```

Parameters

- ♦ **name** The name of the database property whose value you want to obtain. Property names are case insensitive.

Return value

The value of the property as a string.

Remarks

Recognized properties are:

Property	Description
CaseSensitive	The status of the case sensitivity feature. Returns ON if the database is case sensitive. Otherwise, it returns OFF. Database case sensitivity affects how indexes on tables and result sets are sorted. Case sensitivity does not affect how a connection's <code>ULConnectionParms.UserID</code> and <code>ULConnectionParms.Password</code> are verified. User IDs are always case insensitive and passwords are always case sensitive.
CharSet	The character set of the database.
ChecksumLevel	The level of database page checksums enabled for the database.
CollationName	The name of the database's collation sequence.
ConnCount	The number of connections to the database.
date_format	The date format used for string conversions by the database. This format is not necessarily the same as the one used by <code>System.DateTime</code> .

Property	Description
date_order	The date order used for string conversions by the database.
Encryption	The type of encryption applied to the database. Returns None, Simple, AES, or AES_FIPS.
File	The file name of the database.
global_database_id	The value of the global_database_id option used for global autoincrement columns.
MaxHashSize	The default maximum number of bytes to use for index hashing. This property can be set on a per-index basis.
ml_remote_id	The value of the ml_remote_id option used for identifying the database during synchronization.
Name	The name of the database (DBN).
nearest_century	The nearest century used for string conversions by the database.
PageSize	The page size of the database, in bytes.
precision	The floating-point precision used for string conversions by the database.
scale	The minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION during string conversions by the database.
time_format	The time format used for string conversions by the database. This format is not necessarily the same as the one used by System.TimeSpan.
timestamp_format	The timestamp format used for string conversions by the database. This format is not necessarily the same as the one used by System.DateTime.
timestamp_increment	The minimum difference between two unique timestamps, in nanoseconds (1,000,000th of a second).

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“SetDatabaseOption method” on page 654](#)
- ◆ [“CollationName property” on page 647](#)
- ◆ [“UserID property” on page 587](#)
- ◆ [“Password property” on page 586](#)
- ◆ [TimeSpan](#)
- ◆ [DateTime](#)

GetPublicationName method

Returns the name of the publication identified by the specified publication ID. Publication IDs are not publication masks.

Syntax

Visual Basic

```
Public Function GetPublicationName( _  
    ByVal pubID As Integer _  
) As String
```

C#

```
public string GetPublicationName(  
    int pubID  
);
```

Parameters

- ◆ **pubID** The ID of the publication. The value must be in the range [1,PublicationCount].

Return value

The publication name as a string.

Remarks

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“PublicationCount property” on page 648](#)
- ◆ [“PublicationCount property” on page 648](#)

GetPublicationSchema method

Returns the publication schema corresponding to the named publication.

Syntax

Visual Basic

```
Public Function GetPublicationSchema( _  
    ByVal name As String _  
) As ULPublicationSchema
```

C#

```
public ULPublicationSchema GetPublicationSchema(
```

```
    string name
);
```

Parameters

- ◆ **name** The name of the publication.

Return value

The ULPublicationSchema object representing the named publication.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“GetPublicationName method” on page 652](#)
- ◆ [“ULPublicationSchema class” on page 788](#)

GetTableCountInPublications method

Returns the number of tables included in the specified publication mask.

Syntax**Visual Basic**

```
Public Function GetTableCountInPublications( _  
    ByVal mask As Integer _  
) As Integer
```

C#

```
public int GetTableCountInPublications(  
    int mask  
);
```

Parameters

- ◆ **mask** The set of publications to check. For information on building publication masks, see [“ULPublicationSchema class” on page 788](#).

Return value

The number of tables included in set of publications.

Remarks

The count will not include tables whose names end in `_nosync`.

Note: Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs, masks, and counts after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)

GetTableName method

Returns the name of the table identified by the specified table ID.

Syntax

Visual Basic

```
Public Function GetTableName( _  
    ByVal tableID As Integer _  
) As String
```

C#

```
public string GetTableName(  
    int tableID  
);
```

Parameters

- ◆ **tableID** The ID of the table. The value must be in range [1,TableCount].

Return value

The table name as a string.

Remarks

Table IDs may change during a schema upgrade. To correctly identify a table, access it by name or refresh the cached IDs after a schema upgrade.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“TableCount property” on page 649](#)

SetDatabaseOption method

Sets the value for the specified database option.

Syntax

Visual Basic

```
Public Sub SetDatabaseOption( _  
    ByVal name As String, _  
    ByVal value As String _  
)
```

C#

```
public void SetDatabaseOption(  
    string name,  
    string value  
);
```


Parameters

- ◆ **name** The name of the database option. Option names are case insensitive.
- ◆ **value** The new value for the option.

Remarks

Setting a database option results in a commit being performed.

Recognized options are:

Option	Description
global_database_id	The value used for global autoincrement columns. The value must be in the range [0, System.UInt32.MaxValue]. The default is ULConnection.INVALID_DATABASE_ID (used to indicate that the database ID has not been set for the current database).
ml_remote_id	The value used for identifying the database during synchronization. Use a null reference (Nothing in Visual Basic) as the value to remove the ml_remote_id option from the database.

See also

- ◆ [“ULDatabaseSchema class” on page 646](#)
- ◆ [“ULDatabaseSchema members” on page 646](#)
- ◆ [“GetDatabaseProperty method” on page 650](#)
- ◆ [UInt32.MaxValue](#)
- ◆ [“INVALID_DATABASE_ID field” on page 547](#)

ULDataReader class

Represents a read-only bi-directional cursor in an UltraLite database. Cursors are sets of rows from either a table or the result set from a query.

Syntax

Visual Basic

Public Class **ULDataReader**
Inherits DbDataReader

C#

```
public class ULDataReader : DbDataReader
```

Remarks

There is no constructor for ULDataReader. To get a ULDataReader object, execute a ULCommand:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "SELECT emp_id FROM employee FOR READ ONLY", conn _
)
Dim reader As ULDataReader = cmd.ExecuteReader()

// C#
ULCommand cmd = new ULCommand(
    "SELECT emp_id FROM employee FOR READ ONLY", conn
);
ULDataReader reader = cmd.ExecuteReader();
```

UL Ext.: The ADO.NET standard only requires forward-only motion through the result set, but ULDataReader is bi-directional. ULDataReader's Move methods provide you with full flexibility when moving through results.

ULDataReader is a read-only result set. If you need a more flexible object to manipulate results, use a ULCommand.ExecuteResultSet(), ULCommand.ExecuteTable(), or a ULDataAdapter. ULDataReader retrieves rows as needed, whereas ULDataAdapter must retrieve all rows of a result set before you can carry out any action on the object. For large result sets, this difference gives the ULDataReader a much faster response time.

UL Ext.: All columns of a ULDataReader may be retrieved using GetString.

Inherits: System.Data.Common.DbDataReader

Implements: System.Data.IDataReader, System.Data.IDataRecord, System.IDisposable

See also

- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULDataAdapter class” on page 627](#)

- ◆ “GetString method” on page 683
- ◆ DbDataReader
- ◆ IDataReader
- ◆ IDataRecord
- ◆ IDisposable

ULDataReader members

Public properties

Member name	Description
Depth property	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property	Returns the number of columns in the cursor.
HasRows property	Checks whether the ULDataReader has one or more rows.
IsBOF property	UL Ext.: Checks whether the current row position is before the first row.
IsClosed property	Checks whether the cursor is currently open.
IsEOF property	UL Ext.: Checks whether the current row position is after the last row.
Item properties	Gets the value of a specified column as an instance of Object .
RecordsAffected property	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property	UL Ext.: Returns the number of rows in the cursor.
Schema property	UL Ext.: Holds the schema of this cursor.
VisibleFieldCount (inherited from DbDataReader)	

Public methods

Member name	Description
Close method	Closes the cursor.
Dispose (inherited from DbDataReader)	
GetBoolean method	Returns the value for the specified column as a System.Boolean.
GetByte method	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).

Member name	Description
GetBytes methods	UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.
GetChar method	This method is not supported in UltraLite.NET.
GetChars method	Copies a subset of the value for the specified ULDbType.LongVarChar column, beginning at the specified offset, to the specified offset of the destination System.Char array.
GetData (inherited from DbDataReader)	
GetDataTypeName method	Returns the name of the specified column's provider data type.
GetDateTime method	Returns the value for the specified column as a System.DateTime with millisecond accuracy.
GetDecimal method	Returns the value for the specified column as a System.Decimal.
GetDouble method	Returns the value for the specified column as a System.Double.
GetEnumerator method	Returns an System.Collections.IEnumerator that iterates through the ULDataReader.
GetFieldType method	Returns the System.Type most appropriate for the specified column.
GetFloat method	Returns the value for the specified column as a System.Single.
GetGuid method	Returns the value for the specified column as a UUID (System.Guid).
GetInt16 method	Returns the value for the specified column as a System.Int16.
GetInt32 method	Returns the value for the specified column as an Int32.
GetInt64 method	Returns the value for the specified column as an Int64.
GetName method	Returns the name of the specified column.
GetOrdinal method	Returns the column ID of the named column.
GetProviderSpecificFieldType (inherited from DbDataReader)	
GetProviderSpecificValue (inherited from DbDataReader)	
GetProviderSpecificValues (inherited from DbDataReader)	
GetSchemaTable method	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader.

Member name	Description
GetString method	Returns the value for the specified column as a System.String.
GetTimeSpan method	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method	Returns the value for the specified column as a System.UInt16.
GetUInt32 method	Returns the value for the specified column as a UInt32.
GetUInt64 method	Returns the value for the specified column as a System.UInt64.
GetValue method	Returns the value of the specified column in its native format.
GetValues method	Returns all the column values for the current row.
IsDBNull method	Checks whether the value from the specified column is NULL.
MoveAfterLast method	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method	UL Ext.: Positions the cursor to the first row of the cursor.
MoveLast method	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method	UL Ext.: Positions the cursor relative to the current row.
NextResult method	Advances the ULDataReader to the next result when reading the results of batch SQL statements.
Read method	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“GetString method” on page 683](#)
- ◆ [DbDataReader](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

Depth property

Returns the depth of nesting for the current row. The outermost table has a depth of zero.

Syntax

Visual Basic

Public Overrides Readonly Property **Depth** As Integer

C#

public override int **Depth** { get;}

Property value

All UltraLite.NET result sets have a depth of zero.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

FieldCount property

Returns the number of columns in the cursor.

Syntax

Visual Basic

Public Overrides Readonly Property **FieldCount** As Integer

C#

public override int **FieldCount** { get;}

Property value

The number of columns in the cursor as an integer. Returns 0 if the cursor is closed.

Remarks

This method is identical to the `ULCursorSchema.ColumnCount` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“ColumnCount property” on page 619](#)

HasRows property

Checks whether the `ULDataReader` has one or more rows.

Syntax**Visual Basic**

Public Overrides Readonly Property **HasRows** As Boolean

C#

```
public override bool HasRows { get;}
```

Property value

True if the result set has at least one row, false if there are no rows.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

IsBOF property

UL Ext.: Checks whether the current row position is before the first row.

Syntax**Visual Basic**

Public Readonly Property **IsBOF** As Boolean

C#

```
public bool IsBOF { get;}
```

Property value

True if the current row position is before the first row, false otherwise.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

IsClosed property

Checks whether the cursor is currently open.

Syntax**Visual Basic**

Public Overrides Readonly Property **IsClosed** As Boolean

C#

```
public override bool IsClosed { get;}
```

Property value

True if the cursor is currently open, false if the cursor is closed.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

IsEOF property

UL Ext.: Checks whether the current row position is after the last row.

Syntax**Visual Basic**

Public Readonly Property **IsEOF** As Boolean

C#

```
public bool IsEOF { get; }
```

Property value

True if the current row position is after the last row, false otherwise.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

Item properties

Gets the value of a specified column as an instance of [Object](#).

Item(Int32) property

Returns the value of the specified column in its native format. In C#, this property is the indexer for the [ULDataReader](#) class.

Syntax**Visual Basic**

```
Public Overrides Default Readonly Property Item ( _  
    ByVal columnID As Integer _  
) As Object
```

C#

```
public override object this [
```



```
int columnID  
] { get; }
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Property value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

This method is identical in functionality to the ULDataReader.GetValue(int) method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“Item properties” on page 662](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“Item\(String\) property” on page 663](#)
- ◆ [“GetValue method” on page 686](#)
- ◆ [“FieldCount property” on page 660](#)

Item(String) property

Returns the value of the specified named column in its native format. In C#, this property is the indexer for the ULDataReader class.

Syntax**Visual Basic**

```
Public Overrides Default Readonly Property Item ( _  
    ByVal name As String _  
) As Object
```

C#

```
public override object this [  
    string name  
] { get; }
```

Parameters

- ◆ **name** The name of the column.

Property value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

When accessing columns multiple times, it is more efficient to access columns by column ID than by name.

This method is equivalent to:

```
dataReader.GetValue( dataReader.GetOrdinal( name ) )
```

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“Item properties” on page 662](#)
- ◆ [“Item\(Int32\) property” on page 662](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetValue method” on page 686](#)
- ◆ [“GetFieldType method” on page 675](#)

RecordsAffected property

Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.

Syntax

Visual Basic

Public Overrides Readonly Property **RecordsAffected** As Integer

C#

```
public override int RecordsAffected { get;}
```

Property value

The number of rows changed, inserted, or deleted by execution of the SQL statement.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [CommandType.TableDirect](#)

RowCount property

UL Ext.: Returns the number of rows in the cursor.

Syntax

Visual Basic

Public Readonly Property **RowCount** As Integer

C#

```
public int RowCount { get; }
```

Property value

The number of rows in the cursor.

Remarks

One use for RowCount is to decide when to delete old rows to save space. Old rows can be deleted from the UltraLite database without being deleted from the consolidated database using the `ULConnection.StopSynchronizationDelete` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“StartSynchronizationDelete method” on page 572](#)
- ◆ [“StopSynchronizationDelete method” on page 573](#)

Schema property

UL Ext.: Holds the schema of this cursor.

Syntax

Visual Basic

Public Readonly Property **Schema** As ULCursorSchema

C#

```
public ULCursorSchema Schema { get; }
```

Property value

For result sets, the `ULResultSetSchema` object representing the schema of the result set. For tables, the `ULTableSchema` object representing the schema of the table.

Remarks

This property represents the complete schema of the cursor, including UltraLite.NET extended information which is not represented in the results from `ULDataReader.GetSchemaTable`.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“ULTableSchema class” on page 925](#)

- ◆ [“GetSchemaTable method” on page 681](#)
- ◆ [“ULResultSetSchema class” on page 816](#)

Close method

Closes the cursor.

Syntax

Visual Basic

Public Overrides Sub **Close()**

C#

public override void **Close();**

Remarks

It is not an error to close a cursor that is already closed.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

GetBoolean method

Returns the value for the specified column as a System.Boolean.

Syntax

Visual Basic

Public Overrides Function **GetBoolean**(_
 ByVal *columnID* As Integer _
) As Boolean

C#

public override bool **GetBoolean**(
 int *columnID*
);

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Boolean.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Boolean](#)
- ◆ [“FieldCount property” on page 660](#)

GetByte method

Returns the value for the specified column as an unsigned 8-bit value (System.Byte).

Syntax**Visual Basic**

```
Public Overrides Function GetByte( _  
    ByVal columnID As Integer _  
) As Byte
```

C#

```
public override byte GetByte(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Byte.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Byte](#)
- ◆ [“FieldCount property” on page 660](#)

GetBytes methods

UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.

GetBytes(Int32, Int64, Byte[], Int32, Int32) method

Copies a subset of the value for the specified `ULDbType.LongBinary` column, beginning at the specified offset, to the specified offset of the destination `System.Byte` array.

Syntax

Visual Basic

```
Public Overrides Function GetBytes( _  
    ByVal columnID As Integer, _  
    ByVal srcOffset As Long, _  
    ByVal dst As Byte(), _  
    ByVal dstOffset As Integer, _  
    ByVal count As Integer _  
) As Long
```

C#

```
public override long GetBytes(  
    int columnID,  
    long srcOffset,  
    byte[] dst,  
    int dstOffset,  
    int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0, `ULDataReader.FieldCount-1`]. The first column in the cursor has an ID value of zero.
- ◆ **srcOffset** The start position in the column value. Zero is the beginning of the value.
- ◆ **dst** The destination array.
- ◆ **dstOffset** The start position in the destination array.
- ◆ **count** The number of bytes to be copied.

Return value

The actual number of bytes copied.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), `GetBytes` returns the length of the field in bytes.

The bytes at position *srcOffset* through *srcOffset+count-1* of the value are copied into positions *dstOffset* through *dstOffset+count-1*, respectively, of the destination array. If the end of the value is encountered before *count* bytes are copied, the remainder of the destination array is left unchanged.

If any of the following is true, a `ULException` with code `ULSQLCode.SQLE_INVALID_PARAMETER` is thrown and the destination is not modified:

- ◆ *srcOffset* is negative.

- ◆ *dstOffset* is negative.
- ◆ *count* is negative.
- ◆ *dstOffset+count* is greater than *dst.Length*.

For other errors, a *ULException* with the appropriate error code is thrown.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetBytes methods” on page 667](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetBytes\(Int32\) method” on page 669](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [Byte](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“FieldCount property” on page 660](#)

GetBytes(Int32) method

UL Ext.: Returns the value for the specified column as an array of *System.Bytes*. Only valid for columns of type *ULDbType.Binary*, *ULDbType.LongBinary*, or *ULDbType.UniqueIdentifier*.

Syntax

Visual Basic

```
Public Function GetBytes( _  
    ByVal columnID As Integer _  
) As Byte
```

C#

```
public byte GetBytes(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an array of *System.Bytes*.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetBytes methods” on page 667](#)

- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetBytes\(Int32, Int64, Byte\[\], Int32, Int32\) method” on page 668](#)
- ◆ [Byte](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)

GetChar method

This method is not supported in UltraLite.NET.

Syntax

Visual Basic

```
Public Overrides Function GetChar( _  
    ByVal columnID As Integer _  
) As Char
```

C#

```
public override char GetChar(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

This method is not supported in UltraLite.NET.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetString method” on page 683](#)
- ◆ [“FieldCount property” on page 660](#)

GetChars method

Copies a subset of the value for the specified ULDbType.LongVarchar column, beginning at the specified offset, to the specified offset of the destination System.Char array.

Syntax

Visual Basic

```
Public Overrides Function GetChars( _
    ByVal columnID As Integer, _
    ByVal srcOffset As Long, _
    ByVal dst As Char(), _
    ByVal dstOffset As Integer, _
    ByVal count As Integer _
) As Long
```

C#

```
public override long GetChars(
    int columnID,
    long srcOffset,
    char[] dst,
    int dstOffset,
    int count
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **srcOffset** The start position in the column value. Zero is the beginning of the value.
- ◆ **dst** The destination array.
- ◆ **dstOffset** The start position in the destination array.
- ◆ **count** The number of characters to be copied.

Return value

The actual number of characters copied.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), GetChars returns the length of the field in characters.

The characters at position *srcOffset* through *srcOffset+count-1* of the value are copied into positions *dstOffset* through *dstOffset+count-1*, respectively, of the destination array. If the end of the value is encountered before *count* characters are copied, the remainder of the destination array is left unchanged.

If any of the following is true, a *ULException* with code *ULSQLCode.SQLE_INVALID_PARAMETER* is thrown and the destination is not modified:

- ◆ *srcOffset* is negative.
- ◆ *dstOffset* is negative.
- ◆ *count* is negative.
- ◆ *dstOffset+count* is greater than *dst.Length*.

For other errors, a *ULException* with the appropriate error code is thrown.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [Char](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“FieldCount property” on page 660](#)

GetDataTypeName method

Returns the name of the specified column's provider data type.

Syntax**Visual Basic**

```
Public Overrides Function GetDataTypeName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public override string GetDataTypeName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

A string corresponding to the column's ULDbType.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“ULDbType enumeration” on page 695](#)

GetDateTime method

Returns the value for the specified column as a System.DateTime with millisecond accuracy.

Syntax

Visual Basic

```
Public Overrides Function GetDateTime( _  
    ByVal columnID As Integer _  
) As Date
```

C#

```
public override DateTime GetDateTime(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.DateTime.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [DateTime](#)
- ◆ [“FieldCount property” on page 660](#)

GetDecimal method

Returns the value for the specified column as a System.Decimal.

Syntax

Visual Basic

```
Public Overrides Function GetDecimal( _  
    ByVal columnID As Integer _  
) As Decimal
```

C#

```
public override decimal GetDecimal(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Decimal.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Decimal](#)
- ◆ [“FieldCount property” on page 660](#)

GetDouble method

Returns the value for the specified column as a System.Double.

Syntax**Visual Basic**

```
Public Overrides Function GetDouble( _  
    ByVal columnID As Integer _  
) As Double
```

C#

```
public override double GetDouble(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Double.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Double](#)
- ◆ [“FieldCount property” on page 660](#)

GetEnumerator method

Returns an System.Collections.IEnumerator that iterates through the ULDataReader.

Syntax**Visual Basic**

Public Overrides Function **GetEnumerator()** As IEnumerator

C#

public override IEnumerator **GetEnumerator()**;

Return value

A System.Collections.IEnumerator for the ULDataReader.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [IEnumerator](#)

GetFieldType method

Returns the System.Type most appropriate for the specified column.

Syntax**Visual Basic**

Public Overrides Function **GetFieldType**(_
 ByVal *columnID* As Integer _
) As Type

C#

public override Type **GetFieldType**(
 int *columnID*
);

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

A System.Type value for the column.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetDataTypeName method” on page 672](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [Type](#)

- ◆ [“FieldCount property” on page 660](#)

GetFloat method

Returns the value for the specified column as a System.Single.

Syntax

Visual Basic

```
Public Overrides Function GetFloat( _  
    ByVal columnID As Integer _  
) As Single
```

C#

```
public override float GetFloat(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.Single.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Single](#)
- ◆ [“FieldCount property” on page 660](#)

GetGuid method

Returns the value for the specified column as a UUID (System.Guid).

Syntax

Visual Basic

```
Public Overrides Function GetGuid( _  
    ByVal columnID As Integer _  
) As Guid
```

C#

```
public override Guid GetGuid(
```

```
int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a Guid.

Remarks

This method is only valid for columns of type `ULDbType.UniqueIdentifier` or for columns of type `ULDbType.Binary` with length 16.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [“GetColumnSize method” on page 624](#)
- ◆ [Guid](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“FieldCount property” on page 660](#)

GetInt16 method

Returns the value for the specified column as a `System.Int16`.

Syntax**Visual Basic**

```
Public Overrides Function GetInt16( _  
    ByVal columnID As Integer _  
) As Short
```

C#

```
public override short GetInt16(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an System.Int16.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int16](#)
- ◆ [“FieldCount property” on page 660](#)

GetInt32 method

Returns the value for the specified column as an Int32.

Syntax**Visual Basic**

```
Public Overrides Function GetInt32( _  
    ByVal columnID As Integer _  
) As Integer
```

C#

```
public override int GetInt32(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an Int32.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int32](#)
- ◆ [“FieldCount property” on page 660](#)

GetInt64 method

Returns the value for the specified column as an Int64.

Syntax

Visual Basic

```
Public Overrides Function GetInt64( _  
    ByVal columnID As Integer _  
) As Long
```

C#

```
public override long GetInt64(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an Int64.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int64](#)
- ◆ [“FieldCount property” on page 660](#)

GetName method

Returns the name of the specified column.

Syntax

Visual Basic

```
Public Overrides Function GetName( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public override string GetName(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned.

Remarks

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from. For example, MyTable.ID is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

This method is identical to the `ULCursorSchema.GetColumnName` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“GetSchemaTable method” on page 681](#)
- ◆ [“GetColumnName method” on page 621](#)
- ◆ [“FieldCount property” on page 660](#)

GetOrdinal method

Returns the column ID of the named column.

Syntax**Visual Basic**

```
Public Overrides Function GetOrdinal( _  
    ByVal columnName As String _  
) As Integer
```

C#

```
public override int GetOrdinal(  
    string columnName  
);
```

Parameters

- ◆ **columnName** The name of the column.

Return value

The column ID of the named column.

Remarks

Column IDs range from 0 to `ULDataReader.FieldCount-1`, inclusively.

Note that in result sets, not all columns have names and not all column names are unique. If you are not using aliases, the name of a non-computed column is prefixed with the name of the table the column is from.

For example, `MyTable.ID` is the name of the only column in the result set for the query "SELECT ID FROM MyTable".

Column IDs and counts may change during a schema upgrade. To correctly identify a column, access it by name or refresh the cached IDs and counts after a schema upgrade.

This method is identical to the `ULCursorSchema.GetColumnID` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetSchemaTable method” on page 681](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“GetColumnID method” on page 620](#)

GetSchemaTable method

Returns a `System.Data.DataTable` that describes the column metadata of the `ULDataReader`.

Syntax

Visual Basic

Public Overrides Function **GetSchemaTable()** As DataTable

C#

public override DataTable **GetSchemaTable();**

Return value

A `System.Data.DataTable` describing the schema of each column in the `ULDataReader`.

Remarks

The `GetSchemaTable` method returns metadata about each column in the following order:

DataTable column	Description
ColumnName	The name of the column or a null reference (Nothing in Visual Basic) if the column has no name. If the column is aliased in the SQL query, the alias is returned. Note that in result sets, not all columns have names and not all column names are unique.
ColumnOrdinal	The ID of the column. The value is in the range [0,FieldCount-1].
ColumnSize	For sized columns, the maximum length of a value in the column. For other columns, this is the size in bytes of the data type.
NumericPrecision	The precision of a numeric column (<code>ProviderType</code> <code>ULDbType.Decimal</code> or <code>ULDbType.Numeric</code>) or <code>DBNull</code> if the column is not numeric.

DataTable column	Description
NumericScale	The scale of a numeric column (ProviderType ULDbType.Decimal or ULDbType.Numeric) or DBNull if the column is not numeric.
IsUnique	True if the column is a non-computed unique column in the table (BaseTableName) it is taken from.
IsKey	True if the column is one of a set of columns in the result set that taken together from a unique key for the result set. The set of columns with IsKey set to true does not need to be the minimal set that uniquely identifies a row in the result set.
BaseCatalogName	The name of the catalog in the database that contains the column. For UltraLite.NET, this value is always DBNull.
BaseColumnName	The original name of the column in the table BaseTableName of the database or DBNull if the column is computed or if this information cannot be determined.
BaseSchemaName	The name of the schema in the database that contains the column. For UltraLite.NET, this value is always DBNull.
BaseTableName	The name of the table in the database that contains the column, or DBNull if column is computed or if this information cannot be determined.
DataType	The .NET data type that is most appropriate for this type of column.
AllowDBNull	True if the column is nullable, false if the column is not nullable or if this information cannot be determined.
ProviderType	The ULDbType of the column.
IsIdentity	True if the column is an identity column, false if it is not an identity column. For UltraLite.NET, this value is always false.
IsAutoIncrement	True if the column is an autoincrement or global autoincrement column, false otherwise (or if this information cannot be determined).
IsRowVersion	True if the column contains a persistent row identifier that cannot be written to, and has no meaningful value except to identity the row. For UltraLite.NET, this value is always false.
IsLong	True if the column is a ULDbType.LongVarchar or a ULDbType.LongBinary column, false otherwise.
IsReadOnly	True if the column is read-only, false if the column is modifiable or if its access cannot be determined.
IsAliased	True if the column name is an alias, false if it is not an alias.
IsExpression	True if the column is an expression, false if it is a column value.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [DataTable](#)

GetString method

Returns the value for the specified column as a System.String.

Syntax**Visual Basic**

```
Public Overrides Function GetString( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public override string GetString(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.String.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [String](#)
- ◆ [“FieldCount property” on page 660](#)

GetTimeSpan method

Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.

Syntax

Visual Basic

```
Public Function GetTimeSpan( _  
    ByVal columnID As Integer _  
) As TimeSpan
```

C#

```
public TimeSpan GetTimeSpan(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.TimeSpan.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [TimeSpan](#)
- ◆ [“FieldCount property” on page 660](#)

GetUInt16 method

Returns the value for the specified column as a System.UInt16.

Syntax

Visual Basic

```
Public Function GetUInt16( _  
    ByVal columnID As Integer _  
) As UInt16
```

C#

```
public ushort GetUInt16(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an System.UInt16.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt16](#)
- ◆ [“FieldCount property” on page 660](#)

GetUInt32 method

Returns the value for the specified column as a UInt32.

Syntax**Visual Basic**

```
Public Function GetUInt32( _  
    ByVal columnID As Integer _  
) As UInt32
```

C#

```
public uint GetUInt32(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as an UInt32.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt32](#)
- ◆ [“FieldCount property” on page 660](#)

GetUInt64 method

Returns the value for the specified column as a System.UInt64.

Syntax

Visual Basic

```
Public Function GetUInt64( _  
    ByVal columnID As Integer _  
) As UInt64
```

C#

```
public ulong GetUInt64(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as a System.UInt64

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt64](#)
- ◆ [“FieldCount property” on page 660](#)

GetValue method

Returns the value of the specified column in its native format.

Syntax

Visual Basic

```
Public Overrides Function GetValue( _  
    ByVal columnID As Integer _  
) As Object
```

C#

```
public override object GetValue(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

This method is identical in functionality to the `ULDataReader.this[int]`.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“Item\(Int32\) property” on page 662](#)

GetValues method

Returns all the column values for the current row.

Syntax**Visual Basic**

```
Public Overrides Function GetValues( _  
    ByVal values As Object() _  
) As Integer
```

C#

```
public override int GetValues(  
    object[] values  
);
```

Parameters

- ◆ **values** The array of System.Objects to hold the entire row.

Return value

The number of column values retrieved. If the length of the array is greater than the number of columns (`ULDataReader.FieldCount`), only `FieldCount` items are retrieved and the rest of the array is left unchanged.

Remarks

For most applications, the `GetValues` method provides an efficient means for retrieving all columns, rather than retrieving each column individually.

You can pass an `System.Object` array that contains fewer than the number of columns contained in the resulting row. Only the amount of data the `System.Object` array holds is copied to the array. You can also pass an `System.Object` array whose length is more than the number of columns contained in the resulting row.

This method returns DBNull for NULL database columns. For other columns, it returns the value of the column in its native format.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetValue method” on page 686](#)
- ◆ [Object](#)

IsDBNull method

Checks whether the value from the specified column is NULL.

Syntax**Visual Basic**

```
Public Overrides Function IsDBNull( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public override bool IsDBNull(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Return value

True if value is NULL, false if value is not NULL.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“FieldCount property” on page 660](#)

MoveAfterLast method

UL Ext.: Positions the cursor to after the last row of the cursor.

Syntax**Visual Basic**

```
Public Sub MoveAfterLast()
```

C#

```
public void MoveAfterLast();
```

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

MoveBeforeFirst method

UL Ext.: Positions the cursor to before the first row of the cursor.

Syntax**Visual Basic**

```
Public Sub MoveBeforeFirst()
```

C#

```
public void MoveBeforeFirst();
```

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

MoveFirst method

UL Ext.: Positions the cursor to the first row of the cursor.

Syntax**Visual Basic**

```
Public Function MoveFirst() As Boolean
```

C#

```
public bool MoveFirst();
```

Return value

True if successful, false otherwise. For example, the method fails if there are no rows.

See also

- ◆ [“ULDataReader class” on page 656](#)

- ◆ [“ULDataReader members” on page 657](#)

MoveLast method

UL Ext.: Positions the cursor to the last row of the cursor.

Syntax

Visual Basic

Public Function **MoveLast()** As Boolean

C#

public bool **MoveLast();**

Return value

True if successful, false otherwise. For example, the method fails if there are no rows.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

MoveNext method

UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.

Syntax

Visual Basic

Public Function **MoveNext()** As Boolean

C#

public bool **MoveNext();**

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

Remarks

This method is identical to the `ULDataReader.Read` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“Read method” on page 692](#)

MovePrevious method

UL Ext.: Positions the cursor to the previous row or before the first row.

Syntax

Visual Basic

```
Public Function MovePrevious() As Boolean
```

C#

```
public bool MovePrevious();
```

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

MoveRelative method

UL Ext.: Positions the cursor relative to the current row.

Syntax

Visual Basic

```
Public Function MoveRelative( _  
    ByVal offset As Integer _  
) As Boolean
```

C#

```
public bool MoveRelative(  
    int offset  
);
```

Parameters

- ◆ **offset** The number of rows to move. Negative values correspond to moving backward.

Return value

True if successful, false otherwise. For example, the method fails if it positions beyond the first or last row.

Remarks

If the row does not exist, the method returns false, and the cursor position is after the last row (ULDataReader.IsEOF) if

offset is positive, and before the first row (ULDataReader.IsBOF) if the

offset is negative.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“IsEOF property” on page 662](#)
- ◆ [“IsBOF property” on page 661](#)

NextResult method

Advances the ULDataReader to the next result when reading the results of batch SQL statements.

Syntax

Visual Basic

Public Overrides Function **NextResult()** As Boolean

C#

public override bool **NextResult();**

Return value

True if there are more result sets, false otherwise. For UltraLite.NET, always returns false.

Remarks

UL Ext.: UltraLite.NET does not support batches of SQL statements, hence the ULDataReader is always positioned on the first and only result set. Calling NextResult has no effect.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)

Read method

Positions the cursor to the next row, or after the last row if the cursor was already on the last row.

Syntax

Visual Basic

Public Overrides Function **Read()** As Boolean

C#

public override bool **Read();**

Return value

True if successful, false otherwise. For example, the method fails if there are no more rows.

Remarks

This method is identical to the `ULDataReader.MoveNext` method.

See also

- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULDataReader members” on page 657](#)
- ◆ [“MoveNext method” on page 690](#)

ULDateOrder enumeration

UL Ext.: Enumerates the date orders that a database can support.

Syntax

Visual Basic

Public Enum **ULDateOrder**

C#

public enum **ULDateOrder**

Members

Member name	Description	Value
DMY	Day of the month followed by month, followed by year.	2
MDY	Month followed by day of the month, followed by year.	1
YMD	Year followed by month, followed by day of the month.	0

See also

- ◆ [“DateOrder property” on page 611](#)

ULDbType enumeration

Enumerates the UltraLite.NET database data types.

Syntax

Visual Basic

Public Enum **ULDbType**

C#

public enum **ULDbType**

Remarks

The table below lists which .NET types are compatible with each ULDbType. In the case of integral types, table columns can always be set using smaller integer types, but can also be set using larger types as long as the actual value is within the range of the type.

ULDbType	Compatible .NET type	C# built-in type	Visual Basic built-in type
Binary, VarBinary	System.Byte[], or System.Guid if size is 16	byte[]	Byte()
Bit	System.Boolean	bool	Boolean
Char, VarChar	System.String	String	String
Date	System.DateTime	DateTime No built-in type.	Date
Double	System.Double	double	Double
LongBinary	System.Byte[]	byte[]	Byte()
LongVarchar	System.String	String	String
Decimal, Numeric	System.String	decimal	Decimal
Float, Real	System.Single	float	Single
BigInt	System.Int64	long	Long
Integer	System.Int32	int	Integer
SmallInt	System.Int16	short	Short
Time	System.TimeSpan	TimeSpan No built-in type.	TimeSpan No built-in type.

ULDbType	Compatible .NET type	C# built-in type	Visual Basic built-in type
DateTime , TimeStamp	System.DateTime	DateTime No built-in type.	Date
TinyInt	System.Byte	byte	Byte
UnsignedBigInt	System.UInt64	ulong	UInt64 No built-in type.
UnsignedInt	System.UInt32	uint	UInt32 No built-in type.
UnsignedSmallInt	System.UInt16	ushort	UInt16 No built-in type.
UniqueIdentifier	System.Guid	Guid No built-in type.	Guid No built-in type.

Binary columns of length 16 are fully compatible with the UniqueIdentifier type.

Members

Member name	Description	Value
BigInt	Signed 64-bit integer.	5
Binary	Binary data, with a specified maximum length. The enumeration values Binary and VarBinary are aliases of each other.	15
Bit	1-bit flag.	8
Char	Character data, with a specified length. In UltraLite.NET, this type always supports Unicode characters. The types Char and VarChar are fully compatible.	0
Date	Date information.	10
DateTime	Timestamp information (date, time). The enumeration values DateTime and TimeStamp are aliases of each other.	9
Decimal	Exact numerical data, with a specified precision and scale. The enumeration values Decimal and Numeric are aliases of each other.	14
Double	Double precision floating-point number (8 bytes).	12

Member name	Description	Value
Float	Single precision floating-point number (4 bytes). The enumeration values Float and Real are aliases of each other.	13
Integer	Unsigned 32-bit integer.	1
LongBinary	Binary data, with variable length.	18
LongVarchar	Character data, with variable length. In UltraLite.NET, this type always supports Unicode characters.	17
Numeric	Exact numerical data, with a specified precision and scale. The enumeration values Decimal and Numeric are aliases of each other.	14
Real	Single precision floating-point number (4 bytes). The enumeration values Float and Real are aliases of each other.	13
SmallInt	Signed 16-bit integer.	3
Time	Time information.	11
TimeStamp	Timestamp information (date, time). The enumeration values DateTime and TimeStamp are aliases of each other.	9
TinyInt	Unsigned 8-bit integer.	7
UniqueIdentifier	Universally Unique Identifier (UUID/GUID).	19
UnsignedBigInt	Unsigned 64-bit integer.	6
UnsignedInt	Unsigned 32-bit integer.	2
UnsignedSmallInt	Unsigned 16-bit integer.	4
VarBinary	Binary data, with a specified maximum length. The enumeration values Binary and VarBinary are aliases of each other.	15
VarChar	Character data, with a specified maximum length. In UltraLite.NET, this type always supports Unicode characters. The types Char and VarChar are fully compatible.	16

See also

- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetDataTypeName method” on page 672](#)
- ◆ [“GetColumnULDbType method” on page 625](#)
- ◆ [Byte](#)

- ◆ [Guid](#)
- ◆ [Boolean](#)
- ◆ [String](#)
- ◆ [DateTime](#)
- ◆ [Single](#)
- ◆ [Int64](#)
- ◆ [Int32](#)
- ◆ [Int16](#)
- ◆ [TimeSpan](#)
- ◆ [UInt64](#)
- ◆ [UInt32](#)
- ◆ [UInt16](#)

ULException class

Represents a SQL error returned by the UltraLite.NET database. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULException**
Inherits ApplicationException

C#

public sealed class **ULException** : ApplicationException

Remarks

The SQL code denoting the error is returned in the NativeError.

This class is not serializable under the .NET Compact Framework.

See also

- ◆ [“ULException members” on page 699](#)
- ◆ [“NativeError property” on page 701](#)

ULException members

Public constructors

Member name	Description
ULException constructor	Creates a ULException with the given error code.

Public properties

Member name	Description
Data (inherited from Exception)	
HelpLink (inherited from Exception)	
InnerException (inherited from Exception)	
Message (inherited from Exception)	
NativeError property	Returns the SQL code returned by the database.
Source property	Returns the name of the provider that generated the error.

Member name	Description
StackTrace (inherited from Exception)	
TargetSite (inherited from Exception)	

Public methods

Member name	Description
GetBaseException (inherited from Exception)	
GetObjectData method	Populates a SerializationInfo with the data needed to serialize this ULEException.
GetType (inherited from Exception)	
ToString (inherited from Exception)	

See also

- ◆ [“ULEException class” on page 699](#)
- ◆ [“NativeError property” on page 701](#)

ULEException constructor

Creates a ULEException with the given error code.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal code As ULSQLCode, _  
    ByVal s1 As String, _  
    ByVal s2 As String, _  
    ByVal s3 As String _  
)
```

C#

```
public ULEException(  
    ULSQLCode code,  
    string s1,  
    string s2,  
    string s3  
);
```

Parameters

- ◆ **code** The code of the exception.
- ◆ **s1** The first string for the formatted message.
- ◆ **s2** The second string for the formatted message.
- ◆ **s3** The third string for the formatted message.

Remarks

The message string corresponding to the specified `iAnywhere.Data.UltraLite.ULSQLCode` is retrieved from the **`iAnywhere.Data.UltraLite.resources`** assembly. Resources are searched for, by culture, using the following order: `CultureInfo.CurrentUICulture`, then `CultureInfo.CurrentCulture`, and finally culture "EN".

See also

- ◆ [“ULException class” on page 699](#)
- ◆ [“ULException members” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [CultureInfo.CurrentUICulture](#)
- ◆ [CultureInfo.CurrentCulture](#)

NativeError property

Returns the SQL code returned by the database.

Syntax

Visual Basic

Public Readonly Property **NativeError** As ULSQLCode

C#

```
public ULSQLCode NativeError { get;}
```

Property value

The ULSQLCode value returned by the database.

See also

- ◆ [“ULException class” on page 699](#)
- ◆ [“ULException members” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)

Source property

Returns the name of the provider that generated the error.

Syntax**Visual Basic**

Public Overrides Readonly Property **Source** As String

C#

public override string **Source** { get;}

Property value

The string value identifying UltraLite.NET as the provider.

See also

- ◆ [“ULException class” on page 699](#)
- ◆ [“ULException members” on page 699](#)

GetObjectData method

Populates a SerializationInfo with the data needed to serialize this ULException.

Syntax**Visual Basic**

Public Overrides Sub **GetObjectData**(_
 ByVal *info* As SerializationInfo, _
 ByVal *context* As StreamingContext _
)

C#

public override void **GetObjectData**(
 SerializationInfo *info*,
 StreamingContext *context*
);

Parameters

- ◆ **info** The SerializationInfo to populate with data.
- ◆ **context** The destination for this serialization.

Remarks

This method is not supported under the .NET Compact Framework.

See also

- ◆ [“ULException class” on page 699](#)
- ◆ [“ULException members” on page 699](#)

ULFactory class

Represents a set of methods for creating instances of the iAnywhere.Data.UltraLite provider's implementation of the data source classes. This is a static class and so cannot be inherited or instantiated.

Syntax

Visual Basic

Public NotInheritable Class **ULFactory**
Inherits DbProviderFactory

C#

public sealed class **ULFactory** : DbProviderFactory

Remarks

ADO.NET 2.0 adds two new classes, the System.Data.Common.DbProviderFactories class and the System.Data.Common.DbProviderFactory class, to make provider independent code easier to write. To use them with UltraLite.NET specify iAnywhere.Data.UltraLite as the provider invariant name passed to GetFactory. For example:

```
' Visual Basic
Dim factory As DbProviderFactory =
    DbProviderFactories.GetFactory( "iAnywhere.Data.UltraLite" )
Dim conn As DbConnection =
    factory.CreateConnection()

// C#
DbProviderFactory factory =
    DbProviderFactories.GetFactory( "iAnywhere.Data.UltraLite" );
DbConnection conn = factory.CreateConnection();
```

In this example, conn is created as an ULConnection object .

For an explanation of provider factories and generic programming in ADO.NET 2.0, see <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/vsgenerics.asp>.

UltraLite.NET does not support CreateCommandBuilder(), CreateDataSourceEnumerator(), and CreatePermission().

Restrictions: The ULFactory class is not available in the .NET Compact Framework 2.0.

Inherits: System.Data.Common.DbProviderFactory

See also

- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbProviderFactories](#)
- ◆ [DbProviderFactory](#)

ULFactory members

Public fields

Member name	Description
Instance field	Represents the singleton instance of the ULFactory class. This field is read-only.

Public properties

Member name	Description
CanCreateDataSourceEnumerator property	Returns false to indicate the UltraLite.NET does not support DbDataSourceEnumerator.

Public methods

Member name	Description
CreateCommand method	Returns a strongly typed System.Data.Common.DbCommand instance.
CreateCommandBuilder method	Returns a strongly typed System.Data.Common.DbCommandBuilder instance.
CreateConnection method	Returns a strongly typed System.Data.Common.DbConnection instance.
CreateConnectionStringBuilder method	Returns a strongly typed System.Data.Common.DbConnectionStringBuilder instance.
CreateDataAdapter method	Returns a strongly typed System.Data.Common.DbDataAdapter instance.
CreateDataSourceEnumerator (inherited from DbProviderFactory)	
CreateParameter method	Returns a strongly typed System.Data.Common.DbParameter instance.
CreatePermission (inherited from DbProviderFactory)	

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [DbProviderFactories](#)
- ◆ [DbProviderFactory](#)

Instance field

Represents the singleton instance of the ULFactory class. This field is read-only.

Syntax

Visual Basic

Public Shared Readonly **Instance** As ULFactory

C#

```
public const ULFactory Instance ;
```

Remarks

ULFactory is a singleton class, which means only this instance of this class can exist.

Normally you would not use this field directly. Instead, you get a reference to this instance of ULFactory using System.Data.Common.DbProviderFactories.GetFactory(String). For an example, see the [“ULFactory class” on page 703](#).

Restrictions: The ULFactory class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbProviderFactories.GetFactory](#)

CanCreateDataSourceEnumerator property

Returns false to indicate the UltraLite.NET does not support DbDataSourceEnumerator.

Syntax

Visual Basic

Public Overrides Readonly Property **CanCreateDataSourceEnumerator** As Boolean

C#

```
public override bool CanCreateDataSourceEnumerator { get;}
```

Property value

False to indicate that ULFactory does not implement the CreateDataSourceEnumerator() method.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)

CreateCommand method

Returns a strongly typed System.Data.Common.DbCommand instance.

Syntax

Visual Basic

Public Overrides Function **CreateCommand()** As DbCommand

C#

public override DbCommand **CreateCommand();**

Return value

A new ULCommand instance typed as DbCommand.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbCommand](#)
- ◆ [“ULCommand class” on page 497](#)

CreateCommandBuilder method

Returns a strongly typed System.Data.Common.DbCommandBuilder instance.

Syntax

Visual Basic

Public Overrides Function **CreateCommandBuilder()** As DbCommandBuilder

C#

public override DbCommandBuilder **CreateCommandBuilder();**

Return value

A new ULCommandBuilder instance typed as DbCommandBuilder.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbCommandBuilder](#)
- ◆ [“ULCommandBuilder class” on page 532](#)

CreateConnection method

Returns a strongly typed System.Data.Common.DbConnection instance.

Syntax**Visual Basic**

Public Overrides Function **CreateConnection()** As DbConnection

C#

public override DbConnection **CreateConnection();**

Return value

A new ULConnection instance typed as DbConnection.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbConnection](#)
- ◆ [“ULConnection class” on page 542](#)

CreateConnectionStringBuilder method

Returns a strongly typed System.Data.Common.DbConnectionStringBuilder instance.

Syntax**Visual Basic**

Public Overrides Function **CreateConnectionStringBuilder()** As DbConnectionStringBuilder

C#

public override DbConnectionString Builder **CreateConnectionStringBuilder();**

Return value

A new ULConnectionStringBuilder instance typed as DbConnectionStringBuilder.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbConnectionStringBuilder](#)
- ◆ [“ULConnectionStringBuilder class” on page 590](#)

CreateDataAdapter method

Returns a strongly typed System.Data.Common.DbDataAdapter instance.

Syntax

Visual Basic

Public Overrides Function **CreateDataAdapter()** As DbDataAdapter

C#

public override DbDataAdapter **CreateDataAdapter();**

Return value

A new ULDataAdapter instance typed as DbDataAdapter.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbDataAdapter](#)
- ◆ [“ULDataAdapter class” on page 627](#)

CreateParameter method

Returns a strongly typed System.Data.Common.DbParameter instance.

Syntax

Visual Basic

Public Overrides Function **CreateParameter()** As DbParameter

C#

public override DbParameter **CreateParameter();**

Return value

A new ULParameter instance typed as DbParameter.

See also

- ◆ [“ULFactory class” on page 703](#)
- ◆ [“ULFactory members” on page 704](#)
- ◆ [DbParameter](#)
- ◆ [“ULParameter class” on page 751](#)

ULFileTransfer class

UL Ext.: Transfers a file from a remote database using the MobiLink server. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULFileTransfer**

C#

public sealed class **ULFileTransfer**

Remarks

You do not need a database connection to perform a file transfer, however, if your application will be using an UltraLite database with the UltraLite Engine runtime, you must set `ULDatabaseManager.RuntimeType` to the appropriate value before using this API or any other UltraLite.NET API.

To transfer a file you must set the `ULFileTransfer.FileName`, `ULFileTransfer.Stream`, `ULFileTransfer.UserName`, and `ULFileTransfer.Version`.

See also

- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“FileName property” on page 715](#)
- ◆ [“Stream property” on page 717](#)
- ◆ [“UserName property” on page 720](#)
- ◆ [“Version property” on page 720](#)

ULFileTransfer members

Public constructors

Member name	Description
ULFileTransfer constructor	Initializes a <code>ULFileTransfer</code> object. The connection must be opened before you can perform any operations against the database.

Public properties

Member name	Description
AuthStatus property	Returns the authorization status code for the last file transfer attempt.
AuthValue property	Returns the return value from custom user authentication synchronization scripts.
AuthenticationParms property	Specifies parameters for a custom user authentication script (MobiLink <code>authenticate_parameters</code> connection event).

Member name	Description
DestinationFileName property	Specifies the local file name for the downloaded file.
DestinationPath property	Specifies where to download the file.
DownloadedFile property	Checks whether the file was actually downloaded during the last file transfer attempt.
FileAuthCode property	Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.
FileName property	Specifies the name of the file to download.
ForceDownload property	Specifies whether to force the download of the file if it exists.
Password property	The MobiLink password for the user specified by UserName.
ResumePartialDownload property	Specifies whether to resume or discard a previous partial download.
Stream property	Specifies the MobiLink synchronization stream to use for the file transfer.
StreamErrorCode property	Returns the error reported by the stream itself for the last file transfer attempt.
StreamErrorSystem property	Returns the stream error system-specific code.
StreamParms property	Specifies the parameters to configure the synchronization stream.
UserName property	The user name that uniquely identifies the MobiLink client to the MobiLink server.
Version property	Specifies which synchronization script to use.

Public methods

Member name	Description
DownloadFile methods	Download the file specified by the properties of this object.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“FileName property” on page 715](#)
- ◆ [“Stream property” on page 717](#)
- ◆ [“UserName property” on page 720](#)
- ◆ [“Version property” on page 720](#)

ULFileTransfer constructor

Initializes a ULFileTransfer object. The connection must be opened before you can perform any operations against the database.

Syntax

Visual Basic

```
Public Sub New()
```

C#

```
public ULFileTransfer();
```

Remarks

You do not need a database connection to perform a file transfer, however, if your application will be using an UltraLite database with the UltraLite Engine runtime, you must set ULDatabaseManager.RuntimeType to the appropriate value before using this API or any other UltraLite.NET API.

The ULFileTransfer object needs to have the ULFileTransfer.FileName, ULFileTransfer.Stream, ULFileTransfer.UserName, and ULFileTransfer.Version set before it can transfer a file.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)
- ◆ [“RuntimeType property” on page 640](#)
- ◆ [“FileName property” on page 715](#)
- ◆ [“Stream property” on page 717](#)
- ◆ [“UserName property” on page 720](#)
- ◆ [“Version property” on page 720](#)

AuthStatus property

Returns the authorization status code for the last file transfer attempt.

Syntax

Visual Basic

```
Public Readonly Property AuthStatus As ULAAuthStatusCode
```

C#

```
public ULAAuthStatusCode AuthStatus { get;}
```

Property value

One of the ULAAuthStatusCode values denoting the authorization status for the last file transfer attempt.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ULAuthStatusCode enumeration” on page 467](#)

AuthValue property

Returns the return value from custom user authentication synchronization scripts.

Syntax**Visual Basic**

Public Readonly Property **AuthValue** As Long

C#

```
public long AuthValue { get; }
```

Property value

A long integer returned from custom user authentication synchronization scripts.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)

AuthenticationParms property

Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Syntax**Visual Basic**

Public Property **AuthenticationParms** As String

C#

```
public string AuthenticationParms { get; set; }
```

Property value

An array of strings, each containing an authentication parameter (null array entries result in a synchronization error). The default is a null reference (Nothing in Visual Basic), meaning no authentication parameters.

Remarks

Only the first 255 strings are used and each string should be no longer than 128 characters (longer strings are truncated when sent to the MobiLink server).

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)

DestinationFileName property

Specifies the local file name for the downloaded file.

Syntax**Visual Basic**

Public Property **DestinationFileName** As String

C#

public string **DestinationFileName** { get; set; }

Property value

A string specifying the local file name for the downloaded file. If the value is a null reference (Nothing in Visual Basic), FileName is used. The default is a null reference (Nothing in Visual Basic).

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“FileName property” on page 715](#)

DestinationPath property

Specifies where to download the file.

Syntax**Visual Basic**

Public Property **DestinationPath** As String

C#

public string **DestinationPath** { get; set; }

Property value

A string specifying the destination directory of the file. The default is a null reference (Nothing in Visual Basic).

Remarks

The default destination directory varies depending on the device's operating system:

- ◆ For Windows Mobile devices, if the DestinationPath is a null reference (Nothing in Visual Basic), the file is stored in the root (\) directory.
- ◆ For desktop applications, if the DestinationPath is a null reference (Nothing in Visual Basic), the file is stored in the current directory.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ForceDownload property” on page 715](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

DownloadedFile property

Checks whether the file was actually downloaded during the last file transfer attempt.

Syntax**Visual Basic**

Public Readonly Property **DownloadedFile** As Boolean

C#

```
public bool DownloadedFile { get; }
```

Property value

True if the file was downloaded, false otherwise.

Remarks

If the file is already up-to-date when the DownloadFile() is invoked, it will return true, but DownloadedFile will be false. If an error occurs and DownloadFile() returns false, DownloadedFile will be false.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

FileAuthCode property

Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.

Syntax**Visual Basic**

Public Readonly Property **FileAuthCode** As UInt16

C#

```
public ushort FileAuthCode { get; }
```

Property value

An unsigned short integer returned from the `authenticate_file_transfer` script for the last file transfer attempt.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)

FileName property

Specifies the name of the file to download.

Syntax**Visual Basic**

```
Public Property FileName As String
```

C#

```
public string FileName { get; set; }
```

Property value

A string specifying the name of the file as recognized by the MobiLink server. This property has no default value, and must be explicitly set.

Remarks

FileName is the name of the file on the server running MobiLink. MobiLink will first search for this file in the UserName subdirectory and then in the root directory (the root directory is specified via the MobiLink server's `-ftr` option). FileName must not include any drive or path information, or the MobiLink server will be unable to find it. For example, "myfile.txt" is valid, but "somedir\myfile.txt", "..\myfile.txt", and "c:\myfile.txt" are all invalid.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DestinationFileName property” on page 713](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

ForceDownload property

Specifies whether to force the download of the file if it exists.

Syntax

Visual Basic

Public Property **ForceDownload** As Boolean

C#

```
public bool ForceDownload { get; set; }
```

Property value

True to force the download of the file, false to perform normal download checks if file exists. The default is false.

Remarks

If ForceDownload is true, the file will always be downloaded, even if it hasn't changed, and any existing partial download is discarded. If ForceDownload is false, the file will only be downloaded if the server's version of the file is different from the client's. Note that if the file is changed on either the client or the server, specifying ForceDownload true will cause the server version to overwrite the client version.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ResumePartialDownload property” on page 717](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

Password property

The MobiLink password for the user specified by UserName.

Syntax

Visual Basic

Public Property **Password** As String

C#

```
public string Password { get; set; }
```

Property value

A string specifying the MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning no password is specified.

Remarks

The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULFileTransfer class” on page 709](#)

- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“UserName property” on page 720](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

ResumePartialDownload property

Specifies whether to resume or discard a previous partial download.

Syntax

Visual Basic

Public Property **ResumePartialDownload** As Boolean

C#

```
public bool ResumePartialDownload { get; set; }
```

Property value

True to resume a previous partial download, false to discard a previous partial download. The default is false.

Remarks

UltraLite.NET has the ability to restart downloads that fail because of communication errors or user aborts through the ULFileTransferListener. UltraLite.NET processes the download as it is received. If a download is interrupted, then the partially download file is retained and can be resumed during the next file transfer.

If the file has been updated on the server, a partial download will be discarded and a new download started.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ForceDownload property” on page 715](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

Stream property

Specifies the MobiLink synchronization stream to use for the file transfer.

Syntax

Visual Basic

Public Property **Stream** As ULStreamType

C#

```
public ULStreamType Stream { get; set; }
```

Property value

One of the `ULStreamType` values specifying the type of synchronization stream to use. The default is `ULStreamType.TCPIP`.

Remarks

Most synchronization streams require parameters to identify the MobiLink server address and control other behavior. These parameters are supplied by the `ULFileTransfer.StreamParms`.

If the stream type is set to a value that is invalid for the platform, the stream type is set to `ULStreamType.TCPIP`.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ULStreamType enumeration” on page 867](#)
- ◆ [“StreamParms property” on page 719](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)
- ◆ [“ULStreamType enumeration” on page 867](#)

StreamErrorCode property

Returns the error reported by the stream itself for the last file transfer attempt.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorCode** As `ULStreamErrorCode`

C#

```
public ULStreamErrorCode StreamErrorCode { get; }
```

Property value

One of the `ULStreamErrorCode` values denoting the error reported by the stream itself, `ULStreamErrorCode.NONE` if no error occurred.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“ULStreamErrorCode enumeration” on page 846](#)
- ◆ [“ULStreamErrorCode enumeration” on page 846](#)

StreamErrorSystem property

Returns the stream error system-specific code.

Syntax**Visual Basic**

Public Readonly Property **StreamErrorSystem** As Integer

C#

public int **StreamErrorSystem** { get; }

Property value

An integer denoting the stream error system-specific code.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)

StreamParms property

Specifies the parameters to configure the synchronization stream.

Syntax**Visual Basic**

Public Property **StreamParms** As String

C#

public string **StreamParms** { get; set; }

Property value

A string, in the form of a semicolon-separated list of keyword-value pairs, specifying the parameters for the stream. The default is a null reference (Nothing in Visual Basic).

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams” \[MobiLink - Client Administration\]](#).

StreamParms is a string containing all the parameters used for synchronization streams. Parameters are specified as a semicolon-separated list of name=value pairs ("param1=value1;param2=value2").

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“Stream property” on page 717](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)
- ◆ [“ULStreamType enumeration” on page 867](#)

UserName property

The user name that uniquely identifies the MobiLink client to the MobiLink server.

Syntax

Visual Basic

Public Property **UserName** As String

C#

```
public string UserName { get; set; }
```

Property value

A string specifying the user name. This property has no default value, and must be explicitly set.

Remarks

The MobiLink server uses this value to locate the file to download. The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“Password property” on page 716](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

Version property

Specifies which synchronization script to use.

Syntax

Visual Basic

Public Property **Version** As String

C#

```
public string Version { get; set; }
```

Property value

A string specifying the version of the synchronization script to use. This property has no default value, and must be explicitly set.

Remarks

Each synchronization script in the consolidated database is marked with a version string. The version string allows an UltraLite application to choose from a set of synchronization scripts.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DownloadFile\(\) method” on page 721](#)

DownloadFile methods

Download the file specified by the properties of this object.

DownloadFile() method

Download the file specified by the properties of this object.

Syntax**Visual Basic**

Public Function **DownloadFile()** As Boolean

C#

public bool **DownloadFile();**

Return value

True if successful, false otherwise (check ULFileTransfer.StreamErrorCode and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName is downloaded by the MobiLink server to the ULFileTransfer.DestinationPath using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version. Other properties that affect the download are ULFileTransfer.DestinationFileName, ULFileTransfer.AuthenticationParms, ULFileTransfer.ForceDownload and ULFileTransfer.ResumePartialDownload.

To avoid file corruption, UltraLite.NET will download to a temporary file and only replace the destination file once the download has completed.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.DownloadedFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DownloadFile methods” on page 721](#)
- ◆ [“DownloadFile\(ULFileTransferProgressListener\) method” on page 722](#)
- ◆ [“FileName property” on page 715](#)
- ◆ [“DestinationPath property” on page 713](#)

- ◆ “Stream property” on page 717
- ◆ “UserName property” on page 720
- ◆ “Password property” on page 716
- ◆ “Version property” on page 720
- ◆ “DestinationFileName property” on page 713
- ◆ “AuthenticationParms property” on page 712
- ◆ “ForceDownload property” on page 715
- ◆ “ResumePartialDownload property” on page 717
- ◆ “AuthStatus property” on page 711
- ◆ “AuthValue property” on page 712
- ◆ “FileAuthCode property” on page 714
- ◆ “DownloadedFile property” on page 714
- ◆ “StreamErrorCode property” on page 718
- ◆ “StreamErrorSystem property” on page 718
- ◆ “StreamErrorCode property” on page 718

DownloadFile(ULFileTransferProgressListener) method

Download the file specified by the properties of this object with progress events posted to the specified listener.

Syntax

Visual Basic

```
Public Function DownloadFile( _  
    ByVal listener As ULFileTransferProgressListener _  
) As Boolean
```

C#

```
public bool DownloadFile(  
    ULFileTransferProgressListener listener  
);
```

Parameters

- ◆ **listener** The object that receives file transfer progress events.

Return value

True if successful, false otherwise (check ULFileTransfer.StreamErrorCode and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName is downloaded by the MobiLink server to the ULFileTransfer.DestinationPath using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version. Other properties that affect the download are ULFileTransfer.DestinationFileName, ULFileTransfer.AuthenticationParms, ULFileTransfer.ForceDownload and ULFileTransfer.ResumePartialDownload.

To avoid file corruption, UltraLite.NET will download to a temporary file and only replace the destination file once the download has completed.

A detailed result status is reported in this object's `ULFileTransfer.AuthStatus`, `ULFileTransfer.AuthValue`, `ULFileTransfer.FileAuthCode`, `ULFileTransfer.DownloadedFile`, `ULFileTransfer.StreamErrorCode`, and `ULFileTransfer.StreamErrorSystem`.

Errors may result in no data being sent to the listener.

See also

- ◆ [“ULFileTransfer class” on page 709](#)
- ◆ [“ULFileTransfer members” on page 709](#)
- ◆ [“DownloadFile methods” on page 721](#)
- ◆ [“FileName property” on page 715](#)
- ◆ [“DestinationPath property” on page 713](#)
- ◆ [“Stream property” on page 717](#)
- ◆ [“UserName property” on page 720](#)
- ◆ [“Password property” on page 716](#)
- ◆ [“Version property” on page 720](#)
- ◆ [“DestinationFileName property” on page 713](#)
- ◆ [“AuthenticationParms property” on page 712](#)
- ◆ [“ForceDownload property” on page 715](#)
- ◆ [“ResumePartialDownload property” on page 717](#)
- ◆ [“AuthStatus property” on page 711](#)
- ◆ [“AuthValue property” on page 712](#)
- ◆ [“FileAuthCode property” on page 714](#)
- ◆ [“DownloadedFile property” on page 714](#)
- ◆ [“StreamErrorCode property” on page 718](#)
- ◆ [“StreamErrorSystem property” on page 718](#)
- ◆ [“StreamErrorCode property” on page 718](#)

ULFileTransferProgressData class

UL Ext.: Returns file transfer progress monitoring data.

Syntax

Visual Basic

Public Class **ULFileTransferProgressData**

C#

public class **ULFileTransferProgressData**

See also

- ◆ [“ULFileTransferProgressData members” on page 724](#)
- ◆ [“ULFileTransferProgressListener interface” on page 727](#)

ULFileTransferProgressData members

Public fields

Member name	Description
FLAG_IS_BLOCKING field	A flag indicating that the file transfer is blocked awaiting a response from the MobiLink server. This field is constant and read-only.

Public properties

Member name	Description
BytesReceived property	Returns the number of bytes received so far.
FileSize property	Returns the size of the file being transferred.
Flags property	Returns the current file transfer flags indicating additional information relating to the current state.
ResumedAtSize property	Returns the point in the file where the transfer was resumed.

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressListener interface” on page 727](#)

FLAG_IS_BLOCKING field

A flag indicating that the file transfer is blocked awaiting a response from the MobiLink server. This field is constant and read-only.

Syntax**Visual Basic**

Public Shared **FLAG_IS_BLOCKING** As Integer

C#

public const int **FLAG_IS_BLOCKING** ;

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressData members” on page 724](#)

BytesReceived property

Returns the number of bytes received so far.

Syntax**Visual Basic**

Public Readonly Property **BytesReceived** As UInt64

C#

public ulong **BytesReceived** { get;}

Property value

The number of bytes received so far.

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressData members” on page 724](#)

FileSize property

Returns the size of the file being transferred.

Syntax**Visual Basic**

Public Readonly Property **FileSize** As UInt64

C#

public ulong **FileSize** { get;}

Property value

The size of the file in bytes.

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressData members” on page 724](#)

Flags property

Returns the current file transfer flags indicating additional information relating to the current state.

Syntax**Visual Basic**

Public Readonly Property **Flags** As Integer

C#

```
public int Flags { get;}
```

Property value

An integer containing a combination of flags or'ed together.

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressData members” on page 724](#)
- ◆ [“FLAG_IS_BLOCKING field” on page 724](#)

ResumedAtSize property

Returns the point in the file where the transfer was resumed.

Syntax**Visual Basic**

Public Readonly Property **ResumedAtSize** As UInt64

C#

```
public ulong ResumedAtSize { get;}
```

Property value

The number of bytes transferred previously.

See also

- ◆ [“ULFileTransferProgressData class” on page 724](#)
- ◆ [“ULFileTransferProgressData members” on page 724](#)

ULFileTransferProgressListener interface

UL Ext.: The listener interface for receiving file transfer progress events.

Syntax

Visual Basic

Public Interface **ULFileTransferProgressListener**

C#

public interface **ULFileTransferProgressListener**

See also

- ◆ [“ULFileTransferProgressListener members” on page 727](#)
- ◆ [“DownloadFile\(ULFileTransferProgressListener\) method” on page 722](#)

ULFileTransferProgressListener members

Public methods

Member name	Description
FileTransferProgressed method	Invoked during a file transfer to inform the user of progress. This method should return true to cancel the transfer or return false to continue.

See also

- ◆ [“ULFileTransferProgressListener interface” on page 727](#)
- ◆ [“DownloadFile\(ULFileTransferProgressListener\) method” on page 722](#)

FileTransferProgressed method

Invoked during a file transfer to inform the user of progress. This method should return true to cancel the transfer or return false to continue.

Syntax

Visual Basic

Public Function **FileTransferProgressed**(
 ByVal *data* As ULFileTransferProgressData
) As Boolean

C#

public bool **FileTransferProgressed**(
 ULFileTransferProgressData *data*
);

Parameters

- ◆ **data** A ULFileTransferProgressData object containing the latest file transfer progress data.

Return value

This method should return true to cancel the transfer or return false to continue.

Remarks

No UltraLite.NET API methods should be invoked during a FileTransferProgressed call.

See also

- ◆ [“ULFileTransferProgressListener interface” on page 727](#)
- ◆ [“ULFileTransferProgressListener members” on page 727](#)
- ◆ [“ULFileTransferProgressData class” on page 724](#)

ULIndexSchema class

UL Ext.: Represents the schema of an UltraLite table index. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULIndexSchema**

C#

public sealed class **ULIndexSchema**

Remarks

There is no constructor for this class. Index schemas are created using the `ULTableSchema.PrimaryKey`, `ULTableSchema.GetIndex(string)`, and `ULTableSchema.GetOptimalIndex(int)` of the `ULTableSchema`.

See also

- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“PrimaryKey property” on page 928](#)
- ◆ [“GetIndex method” on page 931](#)
- ◆ [“GetOptimalIndex method” on page 932](#)
- ◆ [“ULTableSchema class” on page 925](#)

ULIndexSchema members

Public properties

Member name	Description
ColumnCount property	Returns the number of columns in the index.
IsForeignKey property	Checks whether the index is a foreign key.
IsForeignKeyCheckOnCommit property	Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.
IsForeignKeyNullable property	Checks whether the foreign key is nullable.
IsOpen property	Determines whether the index schema is open or closed.
IsPrimaryKey property	Checks whether the index is the primary key.
IsUniqueIndex property	Checks whether the index is unique.
IsUniqueKey property	Checks whether the index is a unique key.
Name property	Returns the name of the index.

Member name	Description
ReferencedIndexName property	The name of the referenced primary index if the index is a foreign key.
ReferencedTableName property	The name of the referenced primary table if the index is a foreign key.

Public methods

Member name	Description
GetColumnName method	Returns the name of the <i>colOrdinalInIndex</i> 'th column in this index.
IsColumnDescending method	Checks whether the named column is used in descending order by the index.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“PrimaryKey property” on page 928](#)
- ◆ [“GetIndex method” on page 931](#)
- ◆ [“GetOptimalIndex method” on page 932](#)
- ◆ [“ULTableSchema class” on page 925](#)

ColumnCount property

Returns the number of columns in the index.

Syntax

Visual Basic

Public Readonly Property **ColumnCount** As Short

C#

```
public short ColumnCount { get; }
```

Property value

The number of columns in the index.

Remarks

Column ordinals in indexes range from 1 to ColumnCount, inclusive.

Column ordinals and count may change during a schema upgrade. Column ordinals from an index are different than the column IDs in a table or another index, even if they refer to the same physical column in a particular table.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

IsForeignKey property

Checks whether the index is a foreign key.

Syntax**Visual Basic**

Public Readonly Property **IsForeignKey** As Boolean

C#

```
public bool IsForeignKey { get;}
```

Property value

True if the index is the foreign key, false if the index is not the foreign key.

Remarks

Columns in a foreign key may reference another table's non-null, unique index.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

IsForeignKeyCheckOnCommit property

Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.

Syntax**Visual Basic**

Public Readonly Property **IsForeignKeyCheckOnCommit** As Boolean

C#

```
public bool IsForeignKeyCheckOnCommit { get;}
```

Property value

True if referential integrity is checked on commits, false if it is checked on inserts and updates.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“IsForeignKey property” on page 731](#)

IsForeignKeyNullable property

Checks whether the foreign key is nullable.

Syntax

Visual Basic

Public Readonly Property **IsForeignKeyNullable** As Boolean

C#

```
public bool IsForeignKeyNullable { get;}
```

Property value

True if the foreign key is nullable, false if the foreign key is not nullable.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“IsForeignKey property” on page 731](#)

IsOpen property

Determines whether the index schema is open or closed.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

```
public bool IsOpen { get;}
```

Property value

True if the index schema is open, otherwise false.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

IsPrimaryKey property

Checks whether the index is the primary key.

Syntax**Visual Basic**

Public Readonly Property **IsPrimaryKey** As Boolean

C#

```
public bool IsPrimaryKey { get;}
```

Property value

True if the index is the primary key, false if the index is not the primary key.

Remarks

Columns in the primary key may not be null.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

IsUniqueIndex property

Checks whether the index is unique.

Syntax**Visual Basic**

Public Readonly Property **IsUniqueIndex** As Boolean

C#

```
public bool IsUniqueIndex { get;}
```

Property value

True if the index is unique, false if the index is not unique.

Remarks

Columns in a unique index may be null.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

IsUniqueKey property

Checks whether the index is a unique key.

Syntax**Visual Basic**

Public Readonly Property **IsUniqueKey** As Boolean

C#

```
public bool IsUniqueKey { get;}
```

Property value

True if the index is a unique key, false if the index is not a unique key.

Remarks

Columns in a unique key may not be null.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

Name property

Returns the name of the index.

Syntax**Visual Basic**

Public Readonly Property **Name** As String

C#

```
public string Name { get;}
```

Property value

A string specifying the name of the index.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

ReferencedIndexName property

The name of the referenced primary index if the index is a foreign key.

Syntax**Visual Basic**

Public Readonly Property **ReferencedIndexName** As String

C#

```
public string ReferencedIndexName { get;}
```

Property value

A string specifying the name of the referenced primary index.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“IsForeignKey property” on page 731](#)

ReferencedTableName property

The name of the referenced primary table if the index is a foreign key.

Syntax

Visual Basic

```
Public Readonly Property ReferencedTableName As String
```

C#

```
public string ReferencedTableName { get;}
```

Property value

A string specifying the name of the referenced primary table.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“IsForeignKey property” on page 731](#)

GetColumnName method

Returns the name of the

colOrdinalInIndex'th column in this index.

Syntax

Visual Basic

```
Public Function GetColumnName( _  
    ByVal colOrdinalInIndex As Short _  
) As String
```

C#

```
public string GetColumnName(  
    short colOrdinalInIndex  
);
```

Parameters

- ◆ **colOrdinalInIndex** The ordinal of the desired column in the index. The value must be in the range [1,ColumnCount].

Return value

The name of the column.

Remarks

Column ordinals and count may change during a schema upgrade. Column ordinals from an index are different than the column IDs in a table or another index, even if they refer to the same physical column in a particular table.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)
- ◆ [“ColumnCount property” on page 730](#)
- ◆ [“ColumnCount property” on page 730](#)

IsColumnDescending method

Checks whether the named column is used in descending order by the index.

Syntax

Visual Basic

```
Public Function IsColumnDescending(  
    ByVal name As String _  
) As Boolean
```

C#

```
public bool IsColumnDescending(  
    string name  
);
```

Parameters

- ◆ **name** The name of the column.

Return value

True if the column is used in descending order, false if the column is used in ascending order.

See also

- ◆ [“ULIndexSchema class” on page 729](#)
- ◆ [“ULIndexSchema members” on page 729](#)

- ◆ [“GetColumnName method” on page 735](#)
- ◆ [“ColumnCount property” on page 730](#)

ULInfoMessageEventArgs class

Provides data for the ULConnection.InfoMessage event. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULInfoMessageEventArgs**
Inherits EventArgs

C#

public sealed class **ULInfoMessageEventArgs** : EventArgs

See also

- ◆ [“ULInfoMessageEventArgs members” on page 738](#)
- ◆ [“InfoMessage event” on page 575](#)

ULInfoMessageEventArgs members

Public properties

Member name	Description
Message property	The informational or warning message string returned by the database.
NativeError property	The SQL code corresponding to the informational message or warning returned by the database.
Source property	The name of the ADO.NET data provider returning the message.

Public methods

Member name	Description
ToString method	Returns the string representation of the ULConnection.InfoMessage event.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 738](#)
- ◆ [“InfoMessage event” on page 575](#)

Message property

The informational or warning message string returned by the database.

Syntax**Visual Basic**

Public Readonly Property **Message** As String

C#

public string **Message** { get;}

Property value

A string containing the informational or warning message.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 738](#)
- ◆ [“ULInfoMessageEventArgs members” on page 738](#)

NativeError property

The SQL code corresponding to the informational message or warning returned by the database.

Syntax**Visual Basic**

Public Readonly Property **NativeError** As ULSQLCode

C#

public ULSQLCode **NativeError** { get;}

Property value

An informational or warning ULSQLCode value.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 738](#)
- ◆ [“ULInfoMessageEventArgs members” on page 738](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)

Source property

The name of the ADO.NET data provider returning the message.

Syntax**Visual Basic**

Public Readonly Property **Source** As String

C#

```
public string Source { get;}
```

Property value

The string "UltraLite.NET Data Provider".

See also

- ◆ [“ULInfoMessageEventArgs class” on page 738](#)
- ◆ [“ULInfoMessageEventArgs members” on page 738](#)

ToString method

Returns the string representation of the ULConnection.InfoMessage event.

Syntax

Visual Basic

```
Public Overrides Function ToString() As String
```

C#

```
public override string ToString();
```

Return value

The informational or warning message string.

See also

- ◆ [“ULInfoMessageEventArgs class” on page 738](#)
- ◆ [“ULInfoMessageEventArgs members” on page 738](#)
- ◆ [“InfoMessage event” on page 575](#)

ULInfoMessageEventHandler delegate

Represents the method that will handle the `ULConnection.InfoMessage` event.

Syntax

Visual Basic

```
Public Delegate Sub ULInfoMessageEventHandler( _  
    ByVal obj As Object, _  
    ByVal args As ULInfoMessageEventArgs _  
)
```

C#

```
public delegate void ULInfoMessageEventHandler(  
    object obj,  
    ULInfoMessageEventArgs args  
);
```

See also

- ◆ [“InfoMessage event” on page 575](#)
- ◆ [“ULInfoMessageEventArgs class” on page 738](#)

ULMetaDataCollectionNames class

Provides a list of constants for use with the `ULConnection.GetSchema(String,String[])` to retrieve metadata collections. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULMetaDataCollectionNames**

C#

public sealed class **ULMetaDataCollectionNames**

See also

- ♦ [“ULMetaDataCollectionNames members” on page 742](#)
- ♦ [“GetSchema\(String, String\[\]\) method” on page 568](#)

ULMetaDataCollectionNames members

Public properties

Member name	Description
Columns property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the Columns collection.
DataSourceInformation property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the DataSourceInformation collection.
DataTypes property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the DataTypes collection.
ForeignKeys property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the ForeignKeys collection.
IndexColumns property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the IndexColumns collection.
Indexes property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the Indexes collection.
MetaDataCollections property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the MetaDataCollections collection.
Publications property	Provides a list of constants for use with the <code>ULConnection.GetSchema(String,String[])</code> to retrieve metadata collections.
ReservedWords property	Provides a constant for use with the <code>ULConnection.GetSchema(String)</code> that represents the ReservedWords collection.

Member name	Description
Restrictions property	Provides a constant for use with the ULConnection.GetSchema (String) that represents the Restrictions collection.
Tables property	Provides a constant for use with the ULConnection.GetSchema (String) that represents the Tables collection.

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“GetSchema\(String, String\[\]\) method” on page 568](#)

Columns property

Provides a constant for use with the ULConnection.GetSchema(String) that represents the Columns collection.

Syntax**Visual Basic**

Public Shared Readonly Property **Columns** As String

C#

```
public const string Columns { get; }
```

Property value

A string representing the name of the Columns collection.

Example

The following code fills a DataTable with the Columns collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.Columns )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.Columns );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

DataSourceInformation property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the `DataSourceInformation` collection.

Syntax

Visual Basic

Public Shared Readonly Property **DataSourceInformation** As String

C#

public const string **DataSourceInformation** { get;}

Property value

A string representing the name of the `DataSourceInformation` collection.

Example

The following code fills a `DataTable` with the `DataSourceInformation` collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.DataSourceInformation )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.DataSourceInformation );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

DataTypes property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the `DataTypes` collection.

Syntax

Visual Basic

Public Shared Readonly Property **DataTypes** As String

C#

public const string **DataTypes** { get;}

Property value

A string representing the name of the `DataTypes` collection.

Example

The following code fills a DataTable with the DataTypes collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.DataTypes )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.DataTypes );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

ForeignKeys property

Provides a constant for use with the ULConnection.GetSchema(String) that represents the ForeignKeys collection.

Syntax

Visual Basic

Public Shared Readonly Property **ForeignKeys** As String

C#

public const string **ForeignKeys** { get; }

Property value

A string representing the name of the ForeignKeys collection.

Example

The following code fills a DataTable with the ForeignKeys collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.ForeignKeys )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.ForeignKeys );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

IndexColumns property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the `IndexColumns` collection.

Syntax

Visual Basic

Public Shared Readonly Property **IndexColumns** As String

C#

public const string **IndexColumns** { get; }

Property value

A string representing the name of the `IndexColumns` collection.

Example

The following code fills a `DataTable` with the `IndexColumns` collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.IndexColumns )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.IndexColumns );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

Indexes property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the `Indexes` collection.

Syntax

Visual Basic

Public Shared Readonly Property **Indexes** As String

C#

public const string **Indexes** { get; }

Property value

A string representing the name of the `Indexes` collection.

Example

The following code fills a DataTable with the Indexes collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.Indexes )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.Indexes );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

MetaDataCollections property

Provides a constant for use with the ULConnection.GetSchema(String) that represents the MetaDataCollections collection.

Syntax

Visual Basic

Public Shared Readonly Property **MetaDataCollections** As String

C#

public const string **MetaDataCollections** { get; }

Property value

A string representing the name of the MetaDataCollections collection.

Example

The following code fills a DataTable with the MetaDataCollections collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.MetaDataCollections )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.MetaDataCollections );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

Publications property

Provides a list of constants for use with the `ULConnection.GetSchema(String,String[])` to retrieve metadata collections.

Syntax

Visual Basic

Public Shared Readonly Property **Publications** As String

C#

public const string **Publications** { get;}

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String, String\[\]\) method” on page 568](#)

ReservedWords property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the ReservedWords collection.

Syntax

Visual Basic

Public Shared Readonly Property **ReservedWords** As String

C#

public const string **ReservedWords** { get;}

Property value

A string representing the name of the ReservedWords collection.

Example

The following code fills a DataTable with the ReservedWords collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.ReservedWords )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.ReservedWords );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)

- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

Restrictions property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the Restrictions collection.

Syntax

Visual Basic

Public Shared Readonly Property **Restrictions** As String

C#

public const string **Restrictions** { get;}

Property value

A string representing the name of the Restrictions collection.

Example

The following code fills a `DataTable` with the Restrictions collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.Restrictions )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.Restrictions );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

Tables property

Provides a constant for use with the `ULConnection.GetSchema(String)` that represents the Tables collection.

Syntax

Visual Basic

Public Shared Readonly Property **Tables** As String

C#

public const string **Tables** { get;}

Property value

A string representing the name of the Tables collection.

Example

The following code fills a DataTable with the Tables collection.

```
' Visual Basic
Dim schema As DataTable = _
    conn.GetSchema( ULMetaDataCollectionNames.Tables )

// C#
DataTable schema =
    conn.GetSchema( ULMetaDataCollectionNames.Tables );
```

See also

- ◆ [“ULMetaDataCollectionNames class” on page 742](#)
- ◆ [“ULMetaDataCollectionNames members” on page 742](#)
- ◆ [“GetSchema\(String\) method” on page 567](#)

ULParameter class

Represents a parameter to a ULCommand. This class cannot be inherited.

Syntax

Visual Basic

```
Public NotInheritable Class ULParameter  
    Inherits DbParameter  
    Implements ICloneable
```

C#

```
public sealed class ULParameter : DbParameter,  
    ICloneable
```

Remarks

A ULParameter object can be created directly using one of its many constructors, or using the ULCommand.CreateParameter method. Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using the ULParameter(string,object) constructor. For example:

```
' Visual Basic  
Dim p As ULParameter = New ULParameter( "", CType( 0, Object ) )  
  
// C#  
ULParameter p = new ULParameter( "", (object)0 );
```

Parameters (including those created by ULCommand.CreateParameter) must be added to a ULCommand.Parameters collection to be used. All parameters are treated as positional parameters and are used by a command in the order that they were added.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

Inherits: System.Data.Common.DbParameter

Implements: System.Data.IDbDataParameter, System.Data.IDataParameter

See also

- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CreateParameter method” on page 515](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“Value property” on page 765](#)
- ◆ [DbParameter](#)
- ◆ [IDbDataParameter](#)
- ◆ [IDataParameter](#)

ULParameter members

Public constructors

Member name	Description
ULParameter constructors	Initializes a new instance of the “ULParameter class” on page 751.

Public properties

Member name	Description
DbType property	Specifies the System.Data.DbType of the parameter
Direction property	A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.
IsNullable property	Specifies whether the parameter accepts null values.
Offset property	Specifies the offset to the ULParameter.Value.
ParameterName property	Specifies the name of the parameter.
Precision property	Specifies the maximum number of digits used to represent the ULParameter.Value.
Scale property	Specifies the number of decimal places to which ULParameter.Value is resolved.
Size property	Specifies the maximum size of the data within the column.
SourceColumn property	Specifies the name of the source column mapped to the DataSet and used for loading or returning the value.
SourceColumnNullMapping property	
SourceVersion property	The System.Data.DataRowVersion to use when loading ULParameter.Value.
ULDbType property	Specifies the iAnywhere.Data.UltraLite.ULDbType of the parameter
Value property	Specifies the value of the parameter.

Public methods

Member name	Description
ResetDbType method	This method is not supported in UltraLite.NET.
ToString method	Returns the string representation of this instance.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CreateParameter method” on page 515](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [“Value property” on page 765](#)
- ◆ [DbParameter](#)
- ◆ [IDbDataParameter](#)
- ◆ [IDataParameter](#)

ULParameter constructors

Initializes a new instance of the [“ULParameter class” on page 751](#).

ULParameter() constructor

Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

Syntax**Visual Basic**

```
Public Sub New()
```

C#

```
public ULParameter();
```

Example

The following code creates a ULParameter with the value 3 and adds it to a ULCommand called cmd.

```
' Visual Basic
Dim p As ULParameter = New ULParameter
p.Value = 3
cmd.Parameters.Add( p )

// C#
ULParameter p = new ULParameter();
p.Value = 3;
cmd.Parameters.Add( p );
```

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULParameter constructors” on page 753](#)
- ◆ [“Value property” on page 765](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“ULCommand class” on page 497](#)

ULParameter(String, Object) constructor

Initializes a ULParameter object with the specified parameter name and value.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal value As Object _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    object value  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **value** A System.Object that is to be the value of the parameter.

Remarks

Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using this constructor.

Example

The following code creates a ULParameter with the value 0 and adds it to a ULCommand called cmd.

```
' Visual Basic  
cmd.Parameters.Add( New ULParameter( "", CType( 0, Object ) ) )  
  
// C#  
cmd.Parameters.Add( new ULParameter( "", (object)0 ) );
```

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULParameter constructors” on page 753](#)
- ◆ [“ULParameter\(\) constructor” on page 753](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [Object](#)

ULParameter(String, ULDbType) constructor

Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULParameter constructors” on page 753](#)
- ◆ [“ULParameter\(\) constructor” on page 753](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“Value property” on page 765](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDbType enumeration” on page 695](#)

ULParameter(String, ULDbType, Int32) constructor

Initializes a ULParameter object with the specified parameter name and data type. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As UldbType, _  
    ByVal size As Integer _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    UldbType dbType,  
    int size  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.UldbType values.
- ◆ **size** The length of the parameter.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULParameter constructors” on page 753](#)
- ◆ [“ULParameter\(\) constructor” on page 753](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“Value property” on page 765](#)
- ◆ [“UldbType enumeration” on page 695](#)

ULParameter(String, UldbType, Int32, String) constructor

Initializes a ULParameter object with the specified parameter name, data type, and length. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As UldbType, _  
    ByVal size As Integer, _  
    ByVal sourceColumn As String _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType,  
    int size,  
    string sourceColumn  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string (""), or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.
- ◆ **sourceColumn** The name of the source column to map.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULParameter constructors” on page 753](#)
- ◆ [“ULParameter\(\) constructor” on page 753](#)
- ◆ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ◆ [“Value property” on page 765](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULCommand class” on page 497](#)

ULParameter(String, ULDbType, Int32, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object) constructor

Initializes a ULParameter object with the specified parameter name, data type, length, direction, nullability, numeric precision, numeric scale, source column, source version, and value. This constructor is not recommended; it is provided for compatibility with other data providers.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal parameterName As String, _  
    ByVal dbType As ULDbType, _  
    ByVal size As Integer, _  
    ByVal direction As ParameterDirection, _  
    ByVal isNullable As Boolean, _
```

```
ByVal precision As Byte, _  
ByVal scale As Byte, _  
ByVal sourceColumn As String, _  
ByVal sourceVersion As DataRowVersion, _  
ByVal value As Object _  
)
```

C#

```
public ULParameter(  
    string parameterName,  
    ULDbType dbType,  
    int size,  
    ParameterDirection direction,  
    bool isNullable,  
    byte precision,  
    byte scale,  
    string sourceColumn,  
    DataRowVersion sourceVersion,  
    object value  
);
```

Parameters

- ♦ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ♦ **dbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ♦ **size** The length of the parameter.
- ♦ **direction** One of the System.Data.ParameterDirection values.
- ♦ **isNullable** True if the value of the field can be null; otherwise, false.
- ♦ **precision** The total number of digits to the left and right of the decimal point to which Value is resolved.
- ♦ **scale** The total number of decimal places to which Value is resolved.
- ♦ **sourceColumn** The name of the source column to map.
- ♦ **sourceVersion** One of the System.Data.DataRowVersion values.
- ♦ **value** An System.Object that is to be the value of the parameter.

See also

- ♦ [“ULParameter class” on page 751](#)
- ♦ [“ULParameter members” on page 752](#)
- ♦ [“ULParameter constructors” on page 753](#)
- ♦ [“ULParameter\(\) constructor” on page 753](#)
- ♦ [“ULParameter\(String, Object\) constructor” on page 754](#)
- ♦ [“ULDbType enumeration” on page 695](#)
- ♦ [ParameterDirection](#)

- ◆ [DataRowVersion](#)
- ◆ [Object](#)
- ◆ [ParameterDirection.Input](#)
- ◆ [“ULCommand class” on page 497](#)

DbType property

Specifies the System.Data.DbType of the parameter

Syntax

Visual Basic

Public Overrides Property **DbType** As DbType

C#

```
public override DbType DbType { get; set; }
```

Property value

One of the System.Data.DbType values.

Remarks

The ULParameter.ULDbType and DbType properties are linked. Therefore, setting the DbType changes the ULParameter.ULDbType to a supporting iAnywhere.Data.UltraLite.ULDbType.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [DbType](#)
- ◆ [DbType](#)
- ◆ [“ULDbType property” on page 765](#)
- ◆ [“ULDbType enumeration” on page 695](#)

Direction property

A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.

Syntax

Visual Basic

Public Overrides Property **Direction** As ParameterDirection

C#

```
public override ParameterDirection Direction { get; set; }
```

Property value

One of the System.Data.ParameterDirection values.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [ParameterDirection](#)
- ◆ [ParameterDirection.Input](#)
- ◆ [“Value property” on page 765](#)

IsNullable property

Specifies whether the parameter accepts null values.

Syntax**Visual Basic**

Public Overrides Property **IsNullable** As Boolean

C#

```
public override bool IsNullable { get; set; }
```

Property value

True if null values are accepted, false otherwise. The default is false. Null values are handled using the DBNull class.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

Offset property

Specifies the offset to the ULParameter.Value.

Syntax**Visual Basic**

Public Property **Offset** As Integer

C#

```
public int Offset { get; set; }
```

Property value

The offset to the value. The default is 0.

Remarks

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

ParameterName property

Specifies the name of the parameter.

Syntax**Visual Basic**

Public Overrides Property **ParameterName** As String

C#

```
public override string ParameterName { get; set; }
```

Property value

A string representing the name of the parameter, or an empty string ("") for unnamed parameters. Specifying a null reference (Nothing in Visual Basic) results in an empty string being used.

Remarks

In UltraLite.NET, parameter names are not used by ULCommand. All parameters are treated as positional parameters and are used by a command in the order that they were added.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULCommand class” on page 497](#)

Precision property

Specifies the maximum number of digits used to represent the `ULParameter.Value`.

Syntax

Visual Basic

Public Property **Precision** As Byte

C#

public byte **Precision** { get; set; }

Property value

The maximum number of digits used to represent the `ULParameter.Value`. The default value is 0, which indicates that the data provider sets the precision for the `ULParameter.Value`.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the `ULParameter.Value` is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

Scale property

Specifies the number of decimal places to which `ULParameter.Value` is resolved.

Syntax

Visual Basic

Public Property **Scale** As Byte

C#

public byte **Scale** { get; set; }

Property value

The number of decimal places to which `ULParameter.Value` is resolved. The default is 0.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the `ULParameter.Value` is important.

See also

- ◆ [“ULParameter class” on page 751](#)

- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

Size property

Specifies the maximum size of the data within the column.

Syntax

Visual Basic

Public Overrides Property **Size** As Integer

C#

```
public override int Size { get; set; }
```

Property value

The maximum size of the data within the column. The default value is inferred from the parameter value. The Size property is used for binary and string types.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

SourceColumn property

Specifies the name of the source column mapped to the DataSet and used for loading or returning the value.

Syntax

Visual Basic

Public Overrides Property **SourceColumn** As String

C#

```
public override string SourceColumn { get; set; }
```

Property value

A string specifying the name of the source column mapped to the DataSet and used for loading or returning the value.

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the `ULParameter.Value` is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

SourceColumnNullMapping property

Syntax**Visual Basic**

Public Overrides Property **SourceColumnNullMapping** As Boolean

C#

```
public override bool SourceColumnNullMapping { get; set; }
```

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)

SourceVersion property

The `System.Data.DataRowVersion` to use when loading `ULParameter.Value`.

Syntax**Visual Basic**

Public Overrides Property **SourceVersion** As DataRowVersion

C#

```
public override DataRowVersion SourceVersion { get; set; }
```

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)
- ◆ [DataRowVersion](#)

ULDbType property

Specifies the iAnywhere.Data.UltraLite.ULDbType of the parameter

Syntax

Visual Basic

Public Property **ULDbType** As ULDbType

C#

public ULDbType **ULDbType** { get; set; }

Property value

One of the iAnywhere.Data.UltraLite.ULDbType"/> values.

Remarks

The ULDbType and ULParameter.DbType are linked. Therefore, setting the ULDbType changes the ULParameter.DbType to a supporting System.Data.DbType.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)
- ◆ [“DbType property” on page 759](#)
- ◆ [DbType](#)
- ◆ [“ULDbType enumeration” on page 695](#)

Value property

Specifies the value of the parameter.

Syntax

Visual Basic

Public Overrides Property **Value** As Object

C#

public override object **Value** { get; set; }

Property value

A System.Object that specifies the value of the parameter.

Remarks

The value is sent as-is to the data provider without any type conversion or mapping. When the command is executed, the command attempts to convert the value to the required type, signaling a `ULException` with `ULSQLCode.SQLE_CONVERSION_ERROR` if it cannot convert the value.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [Object](#)

ResetDbType method

This method is not supported in UltraLite.NET.

Syntax**Visual Basic**

Public Overrides Sub **ResetDbType()**

C#

public override void **ResetDbType();**

Remarks

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the `ULParameter.Value` is important.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)
- ◆ [“Value property” on page 765](#)

ToString method

Returns the string representation of this instance.

Syntax**Visual Basic**

Public Overrides Function **ToString()** As String

C#

public override string **ToString();**

Return value

The name of the parameter.

See also

- ◆ [“ULParameter class” on page 751](#)
- ◆ [“ULParameter members” on page 752](#)

ULParameterCollection class

Represents all parameters to a ULCommand. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULParameterCollection**
Inherits DbParameterCollection

C#

public sealed class **ULParameterCollection** : DbParameterCollection

Remarks

All parameters in the collection are treated as positional parameters and are specified in the same order as the question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

There is no constructor for ULParameterCollection. You obtain a ULParameterCollection from the ULCommand.Parameters.

Inherits: System.Data.Common.DbParameterCollection

Implements: System.Data.IDataParameterCollection

See also

- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [DbParameterCollection](#)
- ◆ [IDataParameterCollection](#)

ULParameterCollection members

Public properties

Member name	Description
Count property	Returns the number of ULParameter objects in the collection.
IsFixedSize property	Indicates whether the ULParameterCollection has a fixed size.
IsReadOnly property	Indicates whether the ULParameterCollection is read-only.

Member name	Description
IsSynchronized property	Indicates whether the ULParameterCollection is synchronized.
Item properties	Gets and sets a DbParameter in the collection.
SyncRoot property	Returns an object that can be used to synchronize access to the SAPa-parameterCollection.

Public methods

Member name	Description
Add methods	Adds a ULParameter to the collection.
AddRange methods	Adds an array of values to the end of the ULParameterCollection.
Clear method	Removes all the parameters from the collection.
Contains methods	Indicates whether a DbParameter with the specified property exists in the collection.
CopyTo method	Copies ULParameter objects from the ULParameterCollection to the specified array.
GetEnumerator method	Returns an enumerator for the collection.
IndexOf methods	Returns the index of the specified DbParameter object.
Insert method	Inserts an ULParameter in the collection at the specified index.
Remove method	Removes an ULParameter from the collection.
RemoveAt methods	Removes a specified DbParameter object from the collection.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“Parameters property” on page 507](#)
- ◆ [DbParameterCollection](#)
- ◆ [IDataParameterCollection](#)

Count property

Returns the number of ULParameter objects in the collection.

Syntax

Visual Basic

Public Overrides Readonly Property **Count** As Integer

C#

public override int **Count** { get;}

Property value

The number of ULParameter objects in the collection.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

IsFixedSize property

Indicates whether the ULParameterCollection has a fixed size.

Syntax

Visual Basic

Public Overrides Readonly Property **IsFixedSize** As Boolean

C#

public override bool **IsFixedSize** { get;}

Property value

True if this collection has a fixed size, false otherwise.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

IsReadOnly property

Indicates whether the ULParameterCollection is read-only.

Syntax

Visual Basic

Public Overrides Readonly Property **IsReadOnly** As Boolean

C#

public override bool **IsReadOnly** { get;}

Property value

True if this collection is read-only, false otherwise.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

IsSynchronized property

Indicates whether the ULParameterCollection is synchronized.

Syntax**Visual Basic**

Public Overrides Readonly Property **IsSynchronized** As Boolean

C#

```
public override bool IsSynchronized { get; }
```

Property value

True if this collection is synchronized, false otherwise.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

Item properties

Gets and sets a [DbParameter](#) in the collection.

Item(Int32) property

Returns the ULParameter at the specified index. In C#, this property is the indexer for the ULParameterCollection class.

Syntax**Visual Basic**

```
Public Property Item ( _  
    ByVal index As Integer _  
) As ULParameter
```

C#

```
public ULParameter this [
```

```
int index  
] { get; set; }
```

Parameters

- ◆ **index** The zero-based index of the parameter to retrieve. The value must be in the range [0,ULParameterCollection.Count-1]. The first parameter in the collection has an index value of zero.

Property value

The ULParameter at the specified index.

Remarks

This is the strongly-typed version of DbParameterCollection.this[int].

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Item properties” on page 771](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [DbParameterCollection.Item](#)

Item(String) property

Returns the ULParameter with the specified name. In C#, this property is the indexer for the ULParameterCollection class.

Syntax**Visual Basic**

```
Public Property Item ( _  
    ByVal parameterName As String _  
) As ULParameter
```

C#

```
public ULParameter this [  
    string parameterName  
] { get; set; }
```

Parameters

- ◆ **parameterName** The name of the parameter to retrieve.

Property value

The ULParameter with the specified name.

Remarks

This is the strongly-typed version of DbParameterCollection.this[string].

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Item properties” on page 771](#)
- ◆ [“Item\(Int32\) property” on page 662](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetValue method” on page 686](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“ULParameter class” on page 751](#)

SyncRoot property

Returns an object that can be used to synchronize access to the SAParameterCollection.

Syntax**Visual Basic**

Public Overrides Readonly Property **SyncRoot** As Object

C#

public override object **SyncRoot** { get;}

Property value

The object to be used to synchornize access to this collection.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

Add methods

Adds a ULParameter to the collection.

Add(Object) method

Adds a ULParameter to the collection.

Syntax**Visual Basic**

Public Overrides Function **Add**(_
 ByVal *value* As Object _
) As Integer

C#

```
public override int Add(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to add to the collection.

Return value

The index of the new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(ULParameter\) method” on page 774](#)
- ◆ [“Add\(String, Object\) method” on page 775](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ULParameter class” on page 751](#)

Add(ULParameter) method

Adds a ULParameter to the collection.

Syntax**Visual Basic**

```
Public Function Add( _  
    ByVal value As ULParameter _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    ULParameter value  
);
```

Parameters

- ◆ **value** The ULParameter object to add to the collection.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(String, Object\) method” on page 775](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [“CommandText property” on page 503](#)

Add(String, Object) method

Adds a new ULParameter, created using the specified parameter name and value, to the collection.

Syntax**Visual Basic**

```
Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal value As Object _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    object value  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **value** A System.Object that is to be the value of the parameter.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the `ULCommand.CommandText`. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the `ULCommand.CommandText` as there are parameters in the collection. Nulls are substituted for missing parameters.

Because of the special treatment of the 0 and 0.0 constants and the way overloaded methods are resolved, it is highly recommended that you explicitly cast constant values to type object when using this method.

Example

The following code adds a `ULParameter` with the value 0 to a `ULCommand` called `cmd`.

```
' Visual Basic
cmd.Parameters.Add( "", CType( 0, Object ) )

// C#
cmd.Parameters.Add( "", (object)0 );
```

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(ULParameter\) method” on page 774](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [Object](#)

Add(String, ULDbType) method

Adds a new `ULParameter`, created using the specified parameter name and data type, to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal ulDbType As ULDbType _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    ULDbType ulDbType  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(ULParameter\) method” on page 774](#)
- ◆ [“Add\(String, Object\) method” on page 775](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDbType enumeration” on page 695](#)

Add(String, ULDbType, Int32) method

Adds a new ULParameter, created using the specified parameter name, data type, and length, to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal ulDbType As ULDbType, _  
    ByVal size As Integer _  
) As ULParameter
```

C#

```
public ULParameter Add(  
    string parameterName,  
    ULDbType ulDbType,  
    int size  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(ULParameter\) method” on page 774](#)
- ◆ [“Add\(String, Object\) method” on page 775](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDbType enumeration” on page 695](#)

Add(String, ULDbType, Int32, String) method

Adds a new ULParameter, created using the specified parameter name, data type, length, and source column name, to the collection.

Syntax

Visual Basic

```
Public Function Add( _  
    ByVal parameterName As String, _  
    ByVal ulDbType As ULDbType, _  
    ByVal size As Integer, _  
    ByVal sourceColumn As String _  
) As ULParameter
```

C#

```
public ULParameter Add(
```

```
string parameterName,  
ULDbType ulDbType,  
int size,  
string sourceColumn  
);
```

Parameters

- ◆ **parameterName** The name of the parameter. For unnamed parameters, use an empty string ("") or a null reference (Nothing in Visual Basic) for this value. In UltraLite.NET, parameter names are not used by ULCommand.
- ◆ **ulDbType** One of the iAnywhere.Data.UltraLite.ULDbType values.
- ◆ **size** The length of the parameter.
- ◆ **sourceColumn** The name of the source column to map.

Return value

The new ULParameter object.

Remarks

All parameters in the collection are treated as positional parameters and must be added to the collection in the same order as the corresponding question mark placeholders in the ULCommand.CommandText. For example, the first parameter in the collection corresponds to the first question mark in the SQL statement, the second parameter in the collection corresponds to the second question mark in the SQL statement, and so on. There must be at least as many question marks in the ULCommand.CommandText as there are parameters in the collection. Nulls are substituted for missing parameters.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Add methods” on page 773](#)
- ◆ [“Add\(ULParameter\) method” on page 774](#)
- ◆ [“Add\(String, Object\) method” on page 775](#)
- ◆ [“ULParameter class” on page 751](#)
- ◆ [“CommandText property” on page 503](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDbType enumeration” on page 695](#)

AddRange methods

Adds an array of values to the end of the ULParameterCollection.

AddRange(Array) method

Adds an array of values to the end of the ULParameterCollection.

Syntax**Visual Basic**

```
Public Overrides Sub AddRange( _  
    ByVal values As Array _  
)
```

C#

```
public override void AddRange(  
    Array values  
);
```

Parameters

- ♦ **values** An array of ULParameter objects to add to the end of this collection.

See also

- ♦ [“ULParameterCollection class” on page 768](#)
- ♦ [“ULParameterCollection members” on page 768](#)
- ♦ [“AddRange methods” on page 779](#)
- ♦ [“ULParameter class” on page 751](#)

AddRange(ULParameter[]) method

Adds an array of values to the end of the ULParameterCollection.

Syntax**Visual Basic**

```
Public Sub AddRange( _  
    ByVal values As ULParameter() _  
)
```

C#

```
public void AddRange(  
    ULParameter[] values  
);
```

Parameters

- ♦ **values** An array of ULParameter objects to add to the end of this collection.

Remarks

This is the strongly-typed version of DbParameterCollection.AddRange(Array).

See also

- ♦ [“ULParameterCollection class” on page 768](#)
- ♦ [“ULParameterCollection members” on page 768](#)
- ♦ [“AddRange methods” on page 779](#)

- ◆ [“ULParameter class” on page 751](#)
- ◆ [DbParameterCollection.AddRange](#)
- ◆ [“ULParameterCollection class” on page 768](#)

Clear method

Removes all the parameters from the collection.

Syntax

Visual Basic

```
Public Overrides Sub Clear()
```

C#

```
public override void Clear();
```

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

Contains methods

Indicates whether a [DbParameter](#) with the specified property exists in the collection.

Contains(Object) method

Checks whether a ULParameter exists in the collection.

Syntax

Visual Basic

```
Public Overrides Function Contains( _  
    ByVal value As Object _  
) As Boolean
```

C#

```
public override bool Contains(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to check for.

Return value

True if the collection contains the ULParameter, false otherwise.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Contains methods” on page 781](#)
- ◆ [“Contains\(String\) method” on page 782](#)
- ◆ [“ULParameter class” on page 751](#)

Contains(String) method

Checks whether a ULParameter with the specified name exists in the collection.

Syntax**Visual Basic**

```
Public Overrides Function Contains( _  
    ByVal value As String _  
) As Boolean
```

C#

```
public override bool Contains(  
    string value  
);
```

Parameters

- ◆ **value** The name of the parameter to search for.

Return value

True if the collection contains the ULParameter, false otherwise.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“Contains methods” on page 781](#)
- ◆ [“Contains\(Object\) method” on page 781](#)
- ◆ [“ULParameter class” on page 751](#)

CopyTo method

Copies ULParameter objects from the ULParameterCollection to the specified array.

Syntax**Visual Basic**

```
Public Overrides Sub CopyTo( _  
    ByVal array As Array, _  
    ByVal index As Integer _  
)
```


C#

```
public override void CopyTo(  
    Array array,  
    int index  
);
```

Parameters

- ◆ **array** The array into which to copy the ULParameter objects.
- ◆ **index** The starting index of the array.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“ULParameter class” on page 751](#)

GetEnumerator method

Returns an enumerator for the collection.

Syntax

Visual Basic

Public Overrides Function **GetEnumerator()** As IEnumerator

C#

```
public override IEnumerator GetEnumerator();
```

Return value

An ArrayList enumerator enumerating the parameters in the collection.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)

IndexOf methods

Returns the index of the specified [DbParameter](#) object.

IndexOf(Object) method

Returns the location of the ULParameter in the collection.

Syntax**Visual Basic**

```
Public Overrides Function IndexOf( _  
    ByVal value As Object _  
) As Integer
```

C#

```
public override int IndexOf(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to locate.

Return value

The zero-based index of the ULParameter in the collection or -1 if the parameter is not found.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“IndexOf methods” on page 783](#)
- ◆ [“IndexOf\(String\) method” on page 784](#)
- ◆ [“ULParameter class” on page 751](#)

IndexOf(String) method

Returns the location of the ULParameter with the specified name in the collection.

Syntax**Visual Basic**

```
Public Overrides Function IndexOf( _  
    ByVal parameterName As String _  
) As Integer
```

C#

```
public override int IndexOf(  
    string parameterName  
);
```

Parameters

- ◆ **parameterName** The name of the parameter to locate.

Return value

The zero-based index of the ULParameter in the collection or -1 if the parameter is not found.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“IndexOf methods” on page 783](#)
- ◆ [“IndexOf\(Object\) method” on page 783](#)
- ◆ [“ULParameter class” on page 751](#)

Insert method

Inserts an ULParameter in the collection at the specified index.

Syntax**Visual Basic**

```
Public Overrides Sub Insert( _  
    ByVal index As Integer, _  
    ByVal value As Object _  
)
```

C#

```
public override void Insert(  
    int index,  
    object value  
);
```

Parameters

- ◆ **index** The zero-based index where the parameter is to be inserted within the collection.
- ◆ **value** The ULParameter object to insert.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“ULParameter class” on page 751](#)

Remove method

Removes an ULParameter from the collection.

Syntax**Visual Basic**

```
Public Overrides Sub Remove( _  
    ByVal value As Object _  
)
```

C#

```
public override void Remove(  
    object value  
);
```

Parameters

- ◆ **value** The ULParameter object to remove.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“ULParameter class” on page 751](#)

RemoveAt methods

Removes a specified [DbParameter](#) object from the collection.

RemoveAt(Int32) method

Removes the parameter at the specified index in the collection.

Syntax**Visual Basic**

```
Public Overrides Sub RemoveAt(  
    ByVal index As Integer  
)
```

C#

```
public override void RemoveAt(  
    int index  
);
```

Parameters

- ◆ **index** The zero-based index of the parameter to remove. The value must be in the range [0,ULParameterCollection.Count-1]. The first parameter in the collection has an index value of zero.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“RemoveAt methods” on page 786](#)
- ◆ [“RemoveAt\(String\) method” on page 787](#)
- ◆ [“Count property” on page 769](#)

RemoveAt(String) method

Removes the parameter with the specified name from the collection.

Syntax

Visual Basic

```
Public Overrides Sub RemoveAt( _  
    ByVal parameterName As String _  
)
```

C#

```
public override void RemoveAt(  
    string parameterName  
);
```

Parameters

- ◆ **parameterName** The name of the parameter to retrieve.

See also

- ◆ [“ULParameterCollection class” on page 768](#)
- ◆ [“ULParameterCollection members” on page 768](#)
- ◆ [“RemoveAt methods” on page 786](#)
- ◆ [“RemoveAt\(Int32\) method” on page 786](#)

ULPublicationSchema class

UL Ext.: Represents the schema of an UltraLite publication. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULPublicationSchema**

C#

public sealed class **ULPublicationSchema**

Remarks

There is no constructor for this class. Publication schemas are created using the `ULDatabaseSchema.GetPublicationSchema` method of the `ULDatabaseSchema` class.

UltraLite methods requiring a publication mask actually require a set of publications to check. A set is formed by or'ing the publication masks of individual publications. For example:

```
' Visual Basic
Dim mask As Integer = pub1.Mask Or pub2.Mask

// C#
int mask = pub1.Mask | pub2.Mask;
```

Two special mask values are also provided by this class. `ULPublicationSchema.SYNC_ALL_DB` corresponds to the entire database. `ULPublicationSchema.SYNC_ALL_PUBS` corresponds to all publications.

Note: Publication masks may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached masks after a schema upgrade.

See also

- ◆ [“ULPublicationSchema members” on page 789](#)
- ◆ [“Mask property” on page 790](#)
- ◆ [“GetLastDownloadTime method” on page 565](#)
- ◆ [“CountUploadRows\(Int32, UInt32\) method” on page 559](#)
- ◆ [“Schema property” on page 552](#)
- ◆ [“SYNC_ALL_DB field” on page 789](#)
- ◆ [“SYNC_ALL_PUBS field” on page 790](#)
- ◆ [“GetPublicationSchema method” on page 652](#)
- ◆ [“ULDatabaseSchema class” on page 646](#)

ULPublicationSchema members

Public fields

Member name	Description
SYNC_ALL_DB field	Publication mask corresponding to the entire database. This field is constant and read-only.
SYNC_ALL_PUBS field	Publication mask corresponding to all publications. This field is constant and read-only.

Public properties

Member name	Description
IsOpen property	Determines whether the publication schema is open or closed.
Mask property	Returns the publication mask of the publication.
Name property	Returns the name of this publication.

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“Mask property” on page 790](#)
- ◆ [“GetLastDownloadTime method” on page 565](#)
- ◆ [“CountUploadRows\(Int32, UInt32\) method” on page 559](#)
- ◆ [“Schema property” on page 552](#)
- ◆ [“SYNC_ALL_DB field” on page 789](#)
- ◆ [“SYNC_ALL_PUBS field” on page 790](#)
- ◆ [“GetPublicationSchema method” on page 652](#)
- ◆ [“ULDatabaseSchema class” on page 646](#)

SYNC_ALL_DB field

Publication mask corresponding to the entire database. This field is constant and read-only.

Syntax

Visual Basic

Public Shared **SYNC_ALL_DB** As Integer

C#

```
public const int SYNC_ALL_DB ;
```

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“ULPublicationSchema members” on page 789](#)

SYNC_ALL_PUBS field

Publication mask corresponding to all publications. This field is constant and read-only.

Syntax

Visual Basic

Public Shared **SYNC_ALL_PUBS** As Integer

C#

public const int **SYNC_ALL_PUBS** ;

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“ULPublicationSchema members” on page 789](#)

IsOpen property

Determines whether the publication schema is open or closed.

Syntax

Visual Basic

Public Readonly Property **IsOpen** As Boolean

C#

public bool **IsOpen** { get;}

Property value

True if the publication schema is open, false if the publication schema closed.

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“ULPublicationSchema members” on page 789](#)

Mask property

Returns the publication mask of the publication.

Syntax

Visual Basic

Public Readonly Property **Mask** As Integer

C#

```
public int Mask { get;}
```

Property value

The publication mask of the publication.

Remarks

Publication IDs, masks, and counts may change during a schema upgrade. To correctly identify a publication, access it by name, or refresh the cached masks and counts after a schema upgrade.

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“ULPublicationSchema members” on page 789](#)

Name property

Returns the name of this publication.

Syntax**Visual Basic**

Public Readonly Property **Name** As String

C#

```
public string Name { get;}
```

Property value

A string specifying the name of the publication.

See also

- ◆ [“ULPublicationSchema class” on page 788](#)
- ◆ [“ULPublicationSchema members” on page 789](#)

ULResultSet class

UL Ext.: Represents an editable result set in an UltraLite database.

Syntax

Visual Basic

Public Class **ULResultSet**
Inherits ULDataReader

C#

public class **ULResultSet** : ULDataReader

Remarks

There is no constructor for this class. ResultSets are created using the `ULCommand.ExecuteResultSet()` method of the `ULCommand` class.

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
    "SELECT emp_id FROM employee", conn _
)
Dim resultSet As ULResultSet = cmd.ExecuteResultSet()

// C#
ULCommand cmd = new ULCommand(
    "SELECT emp_id FROM employee", conn
);
ULResultSet resultSet = cmd.ExecuteResultSet();
```

A `ULResultSet` object represents an editable result set on which you can perform positioned updates and deletes. For fully editable result sets, use `ULCommand.ExecuteTable()` or a `ULDataAdapter`.

Inherits: `ULDataReader`

Implements: `System.Data.IDataReader`, `System.Data.IDataRecord`, `System.IDisposable`

See also

- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

ULResultSet members

Public properties

Member name	Description
Depth property (inherited from ULDataReader)	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property (inherited from ULDataReader)	Returns the number of columns in the cursor.
HasRows property (inherited from ULDataReader)	Checks whether the ULDataReader has one or more rows.
IsBOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is before the first row.
IsClosed property (inherited from ULDataReader)	Checks whether the cursor is currently open.
IsEOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is after the last row.
Item properties (inherited from ULDataReader)	Gets the value of a specified column as an instance of Object .
RecordsAffected property (inherited from ULDataReader)	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property (inherited from ULDataReader)	UL Ext.: Returns the number of rows in the cursor.
Schema property (inherited from ULDataReader)	UL Ext.: Holds the schema of this cursor.
VisibleFieldCount (inherited from DbDataReader)	

Public methods

Member name	Description
AppendBytes method	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.
AppendChars method	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDbType.LongVarchar column.
Close method (inherited from ULDataReader)	Closes the cursor.
Delete method	Deletes the current row.

Member name	Description
Dispose (inherited from DbDataReader)	
GetBoolean method (inherited from ULDataReader)	Returns the value for the specified column as a System.Boolean.
GetByte method (inherited from ULDataReader)	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
GetBytes methods (inherited from ULDataReader)	UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.
GetChar method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetChars method (inherited from ULDataReader)	Copies a subset of the value for the specified ULDbType.LongVarChar column, beginning at the specified offset, to the specified offset of the destination System.Char array.
GetData (inherited from DbDataReader)	
GetDataTypeName method (inherited from ULDataReader)	Returns the name of the specified column's provider data type.
GetDateTime method (inherited from ULDataReader)	Returns the value for the specified column as a System.DateTime with millisecond accuracy.
GetDecimal method (inherited from ULDataReader)	Returns the value for the specified column as a System.Decimal.
GetDouble method (inherited from ULDataReader)	Returns the value for the specified column as a System.Double.
GetEnumerator method (inherited from ULDataReader)	Returns an System.Collections.IEnumerator that iterates through the ULDataReader.
GetFieldType method (inherited from ULDataReader)	Returns the System.Type most appropriate for the specified column.
GetFloat method (inherited from ULDataReader)	Returns the value for the specified column as a System.Single.
GetGuid method (inherited from ULDataReader)	Returns the value for the specified column as a UUID (System.Guid).
GetInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.Int16.
GetInt32 method (inherited from ULDataReader)	Returns the value for the specified column as an Int32.

Member name	Description
GetInt64 method (inherited from ULDataReader)	Returns the value for the specified column as an Int64.
GetName method (inherited from ULDataReader)	Returns the name of the specified column.
GetOrdinal method (inherited from ULDataReader)	Returns the column ID of the named column.
GetProviderSpecificFieldType (inherited from DbDataReader)	
GetProviderSpecificValue (inherited from DbDataReader)	
GetProviderSpecificValues (inherited from DbDataReader)	
GetSchemaTable method (inherited from ULDataReader)	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader.
GetString method (inherited from ULDataReader)	Returns the value for the specified column as a System.String.
GetTimeSpan method (inherited from ULDataReader)	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt16.
GetUInt32 method (inherited from ULDataReader)	Returns the value for the specified column as a UInt32.
GetUInt64 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt64.
GetValue method (inherited from ULDataReader)	Returns the value of the specified column in its native format.
GetValues method (inherited from ULDataReader)	Returns all the column values for the current row.
IsDBNull method (inherited from ULDataReader)	Checks whether the value from the specified column is NULL.
MoveAfterLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the first row of the cursor.

Member name	Description
MoveLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method (inherited from ULDataReader)	UL Ext.: Positions the cursor relative to the current row.
NextResult method (inherited from ULDataReader)	Advances the ULDataReader to the next result when reading the results of batch SQL statements.
Read method (inherited from ULDataReader)	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
SetBoolean method	Sets the value for the specified column using a System.Boolean .
SetByte method	Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).
SetBytes method	Sets the value for the specified column using an array of System.Bytes .
SetDBNull method	Sets a column to NULL.
SetDateTime method	Sets the value for the specified column using a System.DateTime .
SetDecimal method	Sets the value for the specified column using a System.Decimal .
SetDouble method	Sets the value for the specified column using a System.Double .
SetFloat method	Sets the value for the specified column using a System.Single .
SetGuid method	Sets the value for the specified column using a System.Guid .
SetInt16 method	Sets the value for the specified column using an System.Int16 .
SetInt32 method	Sets the value for the specified column using an System.Int32 .
SetInt64 method	Sets the value for the specified column using an Int64 .
SetString method	Sets the value for the specified column using a System.String .
SetTimeSpan method	Sets the value for the specified column using a System.TimeSpan .
SetToDefault method	Sets the value for the specified column to its default value.
SetUInt16 method	Sets the value for the specified column using a System.UInt16 .
SetUInt32 method	Sets the value for the specified column using an System.UInt32 .

Member name	Description
SetUInt64 method	Sets the value for the specified column using a System.UInt64.
Update method	Updates the current row with the current column values (specified using the set methods).

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ExecuteResultSet\(\) method” on page 526](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULDataAdapter class” on page 627](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

AppendBytes method

Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.

Syntax

Visual Basic

```
Public Sub AppendBytes( _
    ByVal columnID As Integer, _
    ByVal val As Byte(), _
    ByVal srcOffset As Integer, _
    ByVal count As Integer _
)
```

C#

```
public void AppendBytes(
    int columnID,
    byte[] val,
    int srcOffset,
    int count
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The value to append to the current new value for the column.
- ◆ **srcOffset** The start position in the source array.

- ◆ **count** The number of bytes to be copied.

Remarks

The bytes at position *srcOffset* (starting from 0) through *srcOffset+count-1* of the array *val* are appended to the value for the specified column.

When inserting, `ULTable.InsertBegin` initializes the new value to the column's default value. The data in the row is not actually changed until you execute an `ULTable.Insert`, and that change is not made permanent until it is committed.

When updating, the first append on a column clears the current value prior to appending the new value.

If any of the following are true, a `ULException` with code `ULSQLCode.SQLE_INVALID_PARAMETER` is thrown and the destination is not modified:

- ◆ *val* is null.
- ◆ *srcOffset* is negative.
- ◆ *count* is negative.
- ◆ *srcOffset+count* is greater than *val.Length*.

For other errors, a `ULException` with the appropriate error code is thrown.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Byte](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“InsertBegin method” on page 919](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“ULException class” on page 699](#)
- ◆ [“FieldCount property” on page 660](#)

AppendChars method

Appends the specified subset of the specified array of `System.Chars` to the new value for the specified `ULDbType.LongVarchar` column.

Syntax

Visual Basic

```
Public Sub AppendChars( _  
    ByVal columnID As Integer, _  
    ByVal val As Char(), _  
    ByVal srcOffset As Integer, _
```



```
    ByVal count As Integer _  
)
```

C#

```
public void AppendChars(  
    int columnID,  
    char[] val,  
    int srcOffset,  
    int count  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The value to append to the current new value for the column.
- ◆ **srcOffset** The start position in the source array.
- ◆ **count** The number of bytes to be copied.

Remarks

The characters at position *srcOffset* (starting from 0) through *srcOffset+count-1* of the array *val* are appended to the value for the specified column. When inserting, ULTable.InsertBegin initializes the new value to the column's default value. The data in the row is not actually changed until you execute an ULTable.Insert, and that change is not made permanent until it is committed.

When updating, the first append on a column clears the current value prior to appending the new value.

If any of the following is true, a ULException with code ULSQLCode.SQLE_INVALID_PARAMETER is thrown and the destination is not modified:

- ◆ *val* is null.
- ◆ *srcOffset* is negative.
- ◆ *count* is negative.
- ◆ *srcOffset+count* is greater than *value.Length*.

For other errors, a ULException with the appropriate error code is thrown.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Char](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“InsertBegin method” on page 919](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“ULException class” on page 699](#)

- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“FieldCount property” on page 660](#)

Delete method

Deletes the current row.

Syntax

Visual Basic

```
Public Sub Delete()
```

C#

```
public void Delete();
```

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“StartSynchronizationDelete method” on page 572](#)
- ◆ [“StopSynchronizationDelete method” on page 573](#)

SetBoolean method

Sets the value for the specified column using a System.Boolean.

Syntax

Visual Basic

```
Public Sub SetBoolean( _  
    ByVal columnID As Integer, _  
    ByVal val As Boolean _  
)
```

C#

```
public void SetBoolean(  
    int columnID,  
    bool val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Boolean](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetByte method

Sets the value for the specified column using a `System.Byte` (unsigned 8-bit integer).

Syntax

Visual Basic

```
Public Sub SetByte( _  
    ByVal columnID As Integer, _  
    ByVal val As Byte _  
)
```

C#

```
public void SetByte(  
    int columnID,  
    byte val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0, `ULDataReader.FieldCount-1`]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an `ULTable.Insert` or `Update`, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)

- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Byte](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetBytes method

Sets the value for the specified column using an array of System.Bytes.

Syntax

Visual Basic

```
Public Sub SetBytes( _  
    ByVal columnID As Integer, _  
    ByVal val As Byte() _  
)
```

C#

```
public void SetBytes(  
    int columnID,  
    byte[] val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

Only suitable for columns of type ULDbType.Binary or ULDbType.LongBinary, or for columns of type ULDbType.UniqueIdentifier when val is of length 16. The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Byte](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)

- ◆ [“ULDbType enumeration” on page 695](#)

SetDBNull method

Sets a column to NULL.

Syntax

Visual Basic

```
Public Sub SetDBNull( _  
    ByVal columnID As Integer _  
)
```

C#

```
public void SetDBNull(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Remarks

The data is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“IsColumnNullable method” on page 937](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetDateTime method

Sets the value for the specified column using a = "System.DateTime".

Syntax

Visual Basic

```
Public Sub SetDateTime( _  
    ByVal columnID As Integer, _  
    ByVal val As Date _  
)
```

C#

```
public void SetDateTime(  
    int columnID,  
    DateTime val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The set value is accurate to the millisecond. The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [DateTime](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetDecimal method

Sets the value for the specified column using a System.Decimal.

Syntax**Visual Basic**

```
Public Sub SetDecimal( _  
    ByVal columnID As Integer, _  
    ByVal val As Decimal _  
)
```

C#

```
public void SetDecimal(  
    int columnID,  
    decimal val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Decimal](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetDouble method

Sets the value for the specified column using a System.Double.

Syntax

Visual Basic

```
Public Sub SetDouble( _  
    ByVal columnID As Integer, _  
    ByVal val As Double _  
)
```

C#

```
public void SetDouble(  
    int columnID,  
    double val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Double](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetFloat method

Sets the value for the specified column using a System.Single.

Syntax**Visual Basic**

```
Public Sub SetFloat( _  
    ByVal columnID As Integer, _  
    ByVal val As Single _  
)
```

C#

```
public void SetFloat(  
    int columnID,  
    float val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Single](#)
- ◆ [“Insert method” on page 918](#)

- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetGuid method

Sets the value for the specified column using a System.Guid.

Syntax

Visual Basic

```
Public Sub SetGuid( _  
    ByVal columnID As Integer, _  
    ByVal val As Guid _  
)
```

C#

```
public void SetGuid(  
    int columnID,  
    Guid val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed. Only valid for columns of type ULDbType.UniqueIdentifier or for columns of type ULDbType.Binary with length 16.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetNewUUID method” on page 566](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetColumnSize method” on page 624](#)
- ◆ [Guid](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“ULDbType enumeration” on page 695](#)
- ◆ [“FieldCount property” on page 660](#)

SetInt16 method

Sets the value for the specified column using an System.Int16.

Syntax

Visual Basic

```
Public Sub SetInt16( _  
    ByVal columnID As Integer, _  
    ByVal val As Short _  
)
```

C#

```
public void SetInt16(  
    int columnID,  
    short val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int16](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetInt32 method

Sets the value for the specified column using an System.Int32.

Syntax

Visual Basic

```
Public Sub SetInt32( _  
    ByVal columnID As Integer, _
```

```
    ByVal val As Integer _  
)
```

C#

```
public void SetInt32(  
    int columnID,  
    int val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int32](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetInt64 method

Sets the value for the specified column using an Int64.

Syntax**Visual Basic**

```
Public Sub SetInt64( _  
    ByVal columnID As Integer, _  
    ByVal val As Long _  
)
```

C#

```
public void SetInt64(  
    int columnID,  
    long val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [Int64](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetString method

Sets the value for the specified column using a System.String.

Syntax

Visual Basic

```
Public Sub SetString( _  
    ByVal columnID As Integer, _  
    ByVal val As String _  
)
```

C#

```
public void SetString(  
    int columnID,  
    string val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetTimeSpan method

Sets the value for the specified column using a System.TimeSpan.

Syntax**Visual Basic**

```
Public Sub SetTimeSpan( _  
    ByVal columnID As Integer, _  
    ByVal val As TimeSpan _  
)
```

C#

```
public void SetTimeSpan(  
    int columnID,  
    TimeSpan val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The set value is accurate to the millisecond and is normalized to a nonnegative value between 0 and 24 hours. The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [TimeSpan](#)
- ◆ [“Insert method” on page 918](#)

- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetToDefault method

Sets the value for the specified column to its default value.

Syntax

Visual Basic

```
Public Sub SetToDefault( _  
    ByVal columnID As Integer _  
)
```

C#

```
public void SetToDefault(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [“GetColumnDefaultValue method” on page 929](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetUInt16 method

Sets the value for the specified column using a System.UInt16.

Syntax

Visual Basic

```
Public Sub SetUInt16( _
```

```
ByVal columnID As Integer, _  
ByVal val As UInt16 _  
)
```

C#

```
public void SetUInt16(  
    int columnID,  
    ushort val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt16](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetUInt32 method

Sets the value for the specified column using an System.UInt32.

Syntax**Visual Basic**

```
Public Sub SetUInt32( _  
    ByVal columnID As Integer, _  
    ByVal val As UInt32 _  
)
```

C#

```
public void SetUInt32(  
    int columnID,  
    uint val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt32](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

SetUInt64 method

Sets the value for the specified column using a System.UInt64.

Syntax

Visual Basic

```
Public Sub SetUInt64( _  
    ByVal columnID As Integer, _  
    ByVal val As UInt64 _  
)
```

C#

```
public void SetUInt64(  
    int columnID,  
    ulong val  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULDataReader.FieldCount-1]. The first column in the cursor has an ID value of zero.
- ◆ **val** The new value for the column.

Remarks

The data in the row is not actually changed until you execute an ULTable.Insert or Update, and that change is not made permanent until it is committed.

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)
- ◆ [“GetOrdinal method” on page 680](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“GetFieldType method” on page 675](#)
- ◆ [UInt64](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Update method” on page 815](#)
- ◆ [“FieldCount property” on page 660](#)

Update method

Updates the current row with the current column values (specified using the set methods).

Syntax**Visual Basic**

```
Public Sub Update()
```

C#

```
public void Update();
```

See also

- ◆ [“ULResultSet class” on page 792](#)
- ◆ [“ULResultSet members” on page 793](#)

ULResultSetSchema class

UL Ext.: Represents the schema of an UltraLite result set. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULResultSetSchema**
Inherits ULCursorSchema

C#

public sealed class **ULResultSetSchema** : ULCursorSchema

Remarks

There is no constructor for this class. A ULResultSetSchema object is attached to a result set as its ULDataReader.Schema.

A result set schema is only valid while the data reader is open.

Inherits: ULCursorSchema

See also

- ◆ [“ULResultSetSchema members” on page 816](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULResultSetSchema class” on page 816](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“ULCursorSchema class” on page 618](#)

ULResultSetSchema members

Public properties

Member name	Description
ColumnCount property (inherited from ULCursorSchema)	Returns the number of columns in the cursor.
IsOpen property (inherited from ULCursorSchema)	Checks whether the cursor schema is currently open.
Name property	Returns the name of the cursor.

Public methods

Member name	Description
GetColumnID method (inherited from ULCursorSchema)	Returns the column ID of the named column.
GetColumnName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnPrecision method (inherited from ULCursorSchema)	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSQLName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnScale method (inherited from ULCursorSchema)	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSize method (inherited from ULCursorSchema)	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnULDbType method (inherited from ULCursorSchema)	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetSchemaTable method (inherited from ULCursorSchema)	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.

See also

- ◆ [“ULResultSetSchema class” on page 816](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULDataReader class” on page 656](#)
- ◆ [“ULResultSetSchema class” on page 816](#)
- ◆ [“Schema property” on page 665](#)
- ◆ [“ULCursorSchema class” on page 618](#)

Name property

Returns the name of the cursor.

Syntax**Visual Basic**

Public Overrides Readonly Property **Name** As String

C#

```
public override string Name { get; }
```

Property value

The SQL statement that generated the ULResultSetSchema.

See also

- ◆ [“ULResultSetSchema class” on page 816](#)
- ◆ [“ULResultSetSchema members” on page 816](#)

ULRowsCopiedEventArgs class

Represents the set of arguments passed to the ULRowsCopiedEventHandler. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULRowsCopiedEventArgs**

C#

public sealed class **ULRowsCopiedEventArgs**

Remarks

Restrictions: The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULRowsCopiedEventArgs members” on page 819](#)
- ◆ [“ULRowsCopiedEventHandler delegate” on page 822](#)

ULRowsCopiedEventArgs members

Public constructors

Member name	Description
ULRowsCopiedEventArgs constructor	Creates a new instance of the ULRowsCopiedEventArgs object.

Public properties

Member name	Description
Abort property	Gets or sets a value that indicates whether the bulk-copy operation should be aborted.
RowsCopied property	Returns the number of rows copied during the current bulk-copy operation.

See also

- ◆ [“ULRowsCopiedEventArgs class” on page 819](#)
- ◆ [“ULRowsCopiedEventHandler delegate” on page 822](#)

ULRowsCopiedEventArgs constructor

Creates a new instance of the ULRowsCopiedEventArgs object.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal rowsCopied As Long _  
)
```

C#

```
public ULRowsCopiedEventArgs(  
    long rowsCopied  
);
```

Parameters

- ◆ **rowsCopied** An 64-bit integer value that indicates the number of rows copied during the current bulk-copy operation.

Remarks

Restrictions: The `ULRowsCopiedEventArgs` class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULRowsCopiedEventArgs class” on page 819](#)
- ◆ [“ULRowsCopiedEventArgs members” on page 819](#)

Abort property

Gets or sets a value that indicates whether the bulk-copy operation should be aborted.

Syntax

Visual Basic

```
Public Property Abort As Boolean
```

C#

```
public bool Abort { get; set; }
```

Remarks

Restrictions: The `ULRowsCopiedEventArgs` class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULRowsCopiedEventArgs class” on page 819](#)
- ◆ [“ULRowsCopiedEventArgs members” on page 819](#)

RowsCopied property

Returns the number of rows copied during the current bulk-copy operation.

Syntax**Visual Basic**

Public Readonly Property **RowsCopied** As Long

C#

```
public long RowsCopied { get;}
```

Property value

A long integer representing the number of rows copied.

Remarks

Restrictions: The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULRowsCopiedEventArgs class” on page 819](#)
- ◆ [“ULRowsCopiedEventArgs members” on page 819](#)

ULRowsCopiedEventHandler delegate

Represents the method that will handle the ULBulkCopy.ULRowsCopied event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowsCopiedEventHandler( _  
    ByVal sender As Object, _  
    ByVal rowsCopiedEventArgs As ULRowsCopiedEventArgs _  
)
```

C#

```
public delegate void ULRowsCopiedEventHandler(  
    object sender,  
    ULRowsCopiedEventArgs rowsCopiedEventArgs  
);
```

Remarks

Restrictions: The ULRowsCopiedEventHandler delegate is not available in the .NET Compact Framework 2.0.

See also

- ◆ [“ULRowsCopied event” on page 479](#)
- ◆ [Object](#)

ULRowUpdatedEventArgs class

Provides data for the ULDataAdapter.RowUpdated event. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULRowUpdatedEventArgs**
Inherits RowUpdatedEventArgs

C#

public sealed class **ULRowUpdatedEventArgs** : RowUpdatedEventArgs

Remarks

Inherits: System.Data.Common.RowUpdatedEventArgs

See also

- ◆ [“ULRowUpdatedEventArgs members” on page 823](#)
- ◆ [“RowUpdated event” on page 637](#)
- ◆ [RowUpdatedEventArgs](#)

ULRowUpdatedEventArgs members

Public constructors

Member name	Description
ULRowUpdatedEventArgs constructor	Initializes a new instance of the ULRowUpdatedEventArgs class.

Public properties

Member name	Description
Command property	Returns the ULCommand executed when DbDataAdapter.Update (System.Data.DataRow[],System.Data.Common.DataTableMapping) is called.
Errors (inherited from RowUpdatedEventArgs)	
RecordsAffected property	Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement. For SELECT statements this value is -1.
Row (inherited from RowUpdatedEventArgs)	

Member name	Description
RowCount (inherited from RowUpdatedEventArgs)	
StatementType (inherited from RowUpdatedEventArgs)	
Status (inherited from RowUpdatedEventArgs)	
TableMapping (inherited from RowUpdatedEventArgs)	

Public methods

Member name	Description
CopyToRows (inherited from RowUpdatedEventArgs)	Lets you access the rows processed during a batch update operation.

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 823](#)
- ◆ [“RowUpdated event” on page 637](#)
- ◆ [RowUpdatedEventArgs](#)

ULRowUpdatedEventArgs constructor

Initializes a new instance of the [ULRowUpdatedEventArgs](#) class.

Syntax

Visual Basic

```
Public Sub New( _  
    ByVal row As DataRow, _  
    ByVal command As IDbCommand, _  
    ByVal statementType As StatementType, _  
    ByVal tableMapping As DataTableMapping _  
)
```

C#

```
public ULRowUpdatedEventArgs(  
    DataRow row,  
    IDbCommand command,  
    StatementType statementType,  
    DataTableMapping tableMapping  
)
```

Parameters

- ◆ **row** The System.Data.DataRow sent through an DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping).
- ◆ **command** The System.Data.IDbCommand executed when DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping) is called.
- ◆ **statementType** One of the System.Data.StatementType values that specifies the type of query executed.
- ◆ **tableMapping** The System.Data.Common.DataTableMapping sent through an DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping).

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 823](#)
- ◆ [“ULRowUpdatedEventArgs members” on page 823](#)
- ◆ [DataRow](#)
- ◆ [IDbCommand](#)
- ◆ [StatementType](#)
- ◆ [DataTableMapping](#)
- ◆ [DbDataAdapter.Update](#)

Command property

Returns the ULCommand executed when DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping) is called.

Syntax

Visual Basic

Public Readonly Property **Command** As ULCommand

C#

```
public ULCommand Command { get; }
```

Property value

The ULCommand object executed by the update.

Remarks

This is the strongly-typed version of System.Data.Common.RowUpdatedEventArgs.Command.

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 823](#)
- ◆ [“ULRowUpdatedEventArgs members” on page 823](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [RowUpdatedEventArgs.Command](#)

RecordsAffected property

Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement. For SELECT statements this value is -1.

Syntax

Visual Basic

Public Readonly Property **RecordsAffected** As Integer

C#

```
public int RecordsAffected { get;}
```

Property value

The number of rows changed, inserted, or deleted; 0 if no rows were affected or the statement failed; and -1 for SELECT statements.

See also

- ◆ [“ULRowUpdatedEventArgs class” on page 823](#)
- ◆ [“ULRowUpdatedEventArgs members” on page 823](#)

ULRowUpdatedEventHandler delegate

Represents the method that will handle the `ULDataAdapter.RowUpdated` event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatedEventHandler( _  
    ByVal sender As Object, _  
    ByVal e As ULRowUpdatedEventArgs _  
)
```

C#

```
public delegate void ULRowUpdatedEventHandler(  
    object sender,  
    ULRowUpdatedEventArgs e  
);
```

See also

- ◆ [“RowUpdated event” on page 637](#)
- ◆ [“ULRowUpdatedEventArgs class” on page 823](#)

ULRowUpdatingEventArgs class

Provides data for the ULDataAdapter.RowUpdating event. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULRowUpdatingEventArgs**
Inherits RowUpdatingEventArgs

C#

public sealed class **ULRowUpdatingEventArgs** : RowUpdatingEventArgs

Remarks

Inherits: System.Data.Common.RowUpdatingEventArgs

See also

- ◆ [“ULRowUpdatingEventArgs members” on page 828](#)
- ◆ [“RowUpdating event” on page 638](#)
- ◆ [RowUpdatingEventArgs](#)

ULRowUpdatingEventArgs members

Public constructors

Member name	Description
ULRowUpdatingEventArgs constructor	Initializes a new instance of the ULRowUpdatingEventArgs class.

Public properties

Member name	Description
Command property	Specifies the ULCommand to execute when performing the DbDataAdapter.Update(System.Data.DataRow [],System.Data.Common.DataTableMapping).
Errors (inherited from RowUpdatingEventArgs)	
Row (inherited from RowUpdatingEventArgs)	
StatementType (inherited from RowUpdatingEventArgs)	
Status (inherited from RowUpdatingEventArgs)	

Member name	Description
TableMapping (inherited from RowUpdatingEventArgs)	

See also

- ◆ [“ULRowUpdatingEventArgs class” on page 828](#)
- ◆ [“RowUpdating event” on page 638](#)
- ◆ [RowUpdatingEventArgs](#)

ULRowUpdatingEventArgs constructor

Initializes a new instance of the ULRowUpdatingEventArgs class.

Syntax**Visual Basic**

```
Public Sub New( _  
    ByVal row As DataRow, _  
    ByVal command As IDbCommand, _  
    ByVal statementType As StatementType, _  
    ByVal tableMapping As DataTableMapping _  
)
```

C#

```
public ULRowUpdatingEventArgs(  
    DataRow row,  
    IDbCommand command,  
    StatementType statementType,  
    DataTableMapping tableMapping  
);
```

Parameters

- ◆ **row** The System.Data.DataRow to update.
- ◆ **command** The System.Data.IDbCommand to execute during the update.
- ◆ **statementType** One of the System.Data.StatementType values that specifies the type of query executed.
- ◆ **tableMapping** The System.Data.Common.DataTableMapping sent through an DbDataAdapter.Update(System.Data.DataRow[], System.Data.Common.DataTableMapping).

See also

- ◆ [“ULRowUpdatingEventArgs class” on page 828](#)
- ◆ [“ULRowUpdatingEventArgs members” on page 828](#)
- ◆ [DataRow](#)
- ◆ [IDbCommand](#)
- ◆ [StatementType](#)

- ◆ [DataTableMapping](#)
- ◆ [DbDataAdapter.Update](#)

Command property

Specifies the ULCommand to execute when performing the DbDataAdapter.Update(System.Data.DataRow [],System.Data.Common.DataTableMapping).

Syntax

Visual Basic

Public Property **Command** As ULCommand

C#

```
public ULCommand Command { get; set; }
```

Property value

The ULCommand object to execute when updating.

Remarks

This is the strongly-typed version of System.Data.Common.RowUpdatingEventArgs.Command.

See also

- ◆ [“ULRowUpdatingEventArgs class” on page 828](#)
- ◆ [“ULRowUpdatingEventArgs members” on page 828](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [DbDataAdapter.Update](#)
- ◆ [RowUpdatingEventArgs.Command](#)

ULRowUpdatingEventHandler delegate

Represents the method that will handle the `ULDataAdapter.RowUpdating` event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatingEventHandler( _  
    ByVal sender As Object, _  
    ByVal e As ULRowUpdatingEventArgs _  
)
```

C#

```
public delegate void ULRowUpdatingEventHandler(  
    object sender,  
    ULRowUpdatingEventArgs e  
);
```

See also

- ◆ [“RowUpdating event” on page 638](#)
- ◆ [“ULRowUpdatingEventArgs class” on page 828](#)

ULRuntimeType enumeration

UL Ext.: Enumerates the types of UltraLite.NET runtimes.

Syntax

Visual Basic

Public Enum **ULRuntimeType**

C#

public enum **ULRuntimeType**

Members

Member name	Description	Value
STANDALONE_UL	Selects the standalone UltraLite.NET runtime. The standalone runtime accesses databases directly. Databases are accessed more quickly this way, but cannot be shared.	0
UL_ENGINE_CLIENT	Selects the UltraLite engine runtime. The UltraLite.NET engine client communicates with the UltraLite engine to access databases. This means that databases can be shared by different applications.	1

See also

- ◆ [“RuntimeType property” on page 640](#)

ULServerSyncListener interface

UL Ext.: The listener interface for receiving server synchronization messages.

Syntax

Visual Basic

Public Interface **ULServerSyncListener**

C#

public interface **ULServerSyncListener**

See also

- ◆ [“ULServerSyncListener members” on page 833](#)

ULServerSyncListener members

Public methods

Member name	Description
ServerSyncInvoked method	Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

See also

- ◆ [“ULServerSyncListener interface” on page 833](#)

ServerSyncInvoked method

Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

Syntax

Visual Basic

```
Public Sub ServerSyncInvoked( _  
    ByVal messageName As String _  
)
```

C#

```
public void ServerSyncInvoked(  
    string messageName  
);
```

Parameters

- ♦ **messageName** The name of the message sent to the application.

Remarks

This method is invoked by a separate thread. To avoid multi-threading issues, it should post an event to the UI. If you are using multi-threading, it is recommended that you use a separate connection and use the lock keyword to access any objects shared with the rest of the application.

Example

The following code fragments demonstrate how to receive a server synchronization request and perform a synchronization in the UI thread.

```
' Visual Basic
Imports iAnywhere.Data.UltraLite

Public Class MainWindow
    Inherits System.Windows.Forms.Form
    Implements ULServerSyncListener
    Private conn As ULConnection

    Public Sub New(ByVal args() As String)

        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        ULConnection.DatabaseManager.SetServerSyncListener( _
            "myCompany.mymsg", "myCompany.myapp", Me _
        )
        'Create Connection
        ...
    End Sub

    Protected Overrides Sub OnClosing( _
        ByVal e As System.ComponentModel.CancelEventArgs _
    )
        ULConnection.DatabaseManager.SetServerSyncListener( _
            Nothing, Nothing, Nothing _
        )
        MyBase.OnClosing(e)
    End Sub

    Public Sub ServerSyncInvoked(ByVal messageName As String) _
        Implements ULServerSyncListener.ServerSyncInvoked
        Me.Invoke(New EventHandler(AddressOf Me.ServerSyncAction))
    End Sub

    Public Sub ServerSyncAction( _
        ByVal sender As Object, ByVal e As EventArgs _
    )
        ' Do Server sync
        conn.Synchronize()
    End Sub
End Class

// C#
```

```
using iAnywhere.Data.UltraLite;
public class Form1 : System.Windows.Forms.Form, ULServerSyncListener
{
    private System.Windows.Forms.MainMenu mainMenu1;
    private ULConnection conn;

    public Form1()
    {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after
        // InitializeComponent call
        //
        ULConnection.DatabaseManager.SetServerSyncListener(
            "myCompany.mymsg", "myCompany.myapp", this
        );
        // Create connection
        ...
    }

    protected override void Dispose( bool disposing )
    {
        base.Dispose( disposing );
    }

    protected override void OnClosing(
        System.ComponentModel.CancelEventArgs e
    )
    {
        ULConnection.DatabaseManager.SetServerSyncListener(
            null, null, null
        );
        base.OnClosing(e);
    }

    public void ServerSyncInvoked( string messageName )
    {
        this.Invoke( new EventHandler( ServerSyncHandler ) );
    }

    internal void ServerSyncHandler(object sender, EventArgs e)
    {
        conn.Synchronize();
    }
}
```

See also

- ◆ [“ULServerSyncListener interface” on page 833](#)
- ◆ [“ULServerSyncListener members” on page 833](#)

ULSQLCode enumeration

UL Ext.: Enumerates the SQL codes that may be reported by UltraLite.NET.

Syntax

Visual Basic

Public Enum **ULSQLCode**

C#

public enum **ULSQLCode**

Members

Member name	Description	Value
SQLE_AGGREGATE_S_NOT_ALLOWED	See “Invalid use of an aggregate function” [SQL Anywhere 10 - Error Messages].	-150
SQLE_ALIAS_NOT_UNIQUE	See “Alias '%1' is not unique” [SQL Anywhere 10 - Error Messages].	-830
SQLE_ALIAS_NOT_YET_DEFINED	See “Definition for alias '%1' must appear before its first reference” [SQL Anywhere 10 - Error Messages].	-831
SQLE_AMBIGUOUS_INDEX_NAME	See “Index name '%1' is ambiguous” [SQL Anywhere 10 - Error Messages].	-678
SQLE_ARGUMENT_CANNOT_BE_NULL	See “Argument %1 of procedure '%2' cannot be NULL” [SQL Anywhere 10 - Error Messages].	-90
SQLE_BAD_ENCRYPTION_KEY	See “Incorrect or missing encryption key” [SQL Anywhere 10 - Error Messages].	-840
SQLE_BAD_PARAMETER_INDEX	See “Input parameter index out of range” [SQL Anywhere 10 - Error Messages].	-689
SQLE_CANNOT_ACCESS_FILESYSTEM	See “Unable to access the filesystem on the device” [SQL Anywhere 10 - Error Messages].	-1108
SQLE_CANNOT_CHANGE_ML_REMOTE_ID	See “Cannot change the MobiLink remote id when the status of the last upload is unknown” [SQL Anywhere 10 - Error Messages].	-1118
SQLE_CANNOT_EXECUTE_STMT	See “Statement cannot be executed” [SQL Anywhere 10 - Error Messages].	111
SQLE_CANNOT_MODIFY	See “Cannot modify column '%1' in table '%2’” [SQL Anywhere 10 - Error Messages].	-191
SQLE_CANNOT_REGISTER_LISTENER	See “The specified listener could not be registered” [SQL Anywhere 10 - Error Messages].	-992

Member name	Description	Value
SQLE_CLIENT_OUT_OF_MEMORY	See “Client out of memory” [<i>SQL Anywhere 10 - Error Messages</i>].	-876
SQLE_COLUMN_AMBIGUOUS	See “Column '%1' found in more than one table -- need a correlation name” [<i>SQL Anywhere 10 - Error Messages</i>].	-144
SQLE_COLUMN_CANT_BE_NULL	See “Column '%1' in table '%2' cannot be NULL” [<i>SQL Anywhere 10 - Error Messages</i>].	-195
SQLE_COLUMN_IN_INDEX	See “Cannot alter a column in an index” [<i>SQL Anywhere 10 - Error Messages</i>].	-127
SQLE_COLUMN_NOT_FOUND	See “Column '%1' not found” [<i>SQL Anywhere 10 - Error Messages</i>].	-143
SQLE_COLUMN_NOT_INDEXED	See “Column '%1' not part of any indexes in its containing table” [<i>SQL Anywhere 10 - Error Messages</i>].	-1101
SQLE_COLUMN_NOT_STREAMABLE	See “The operation failed because column '%1's type does not support streaming” [<i>SQL Anywhere 10 - Error Messages</i>].	-1100
SQLE_COMMUNICATIONS_ERROR	See “Communication error” [<i>SQL Anywhere 10 - Error Messages</i>].	-85
SQLE_CONNECTION_ALREADY_EXISTS	See “This connection already exists” [<i>SQL Anywhere 10 - Error Messages</i>].	-955
SQLE_CONNECTION_NOT_FOUND	See “Connection not found” [<i>SQL Anywhere 10 - Error Messages</i>].	-108
SQLE_CONNECTION_RESTORED	See “UltraLite connection was restored” [<i>SQL Anywhere 10 - Error Messages</i>].	133
SQLE_CONSTRAINT_NOT_FOUND	See “Constraint '%1' not found” [<i>SQL Anywhere 10 - Error Messages</i>].	-929
SQLE_CONVERSION_ERROR	See “Cannot convert %1 to a %2” [<i>SQL Anywhere 10 - Error Messages</i>].	-157
SQLE_COULD_NOT_FIND_FUNCTION	See “Could not find '%1' in dynamic library '%2’” [<i>SQL Anywhere 10 - Error Messages</i>].	-621
SQLE_COULD_NOT_LOAD_LIBRARY	See “Could not load dynamic library '%1’” [<i>SQL Anywhere 10 - Error Messages</i>].	-620
SQLE_CURSOR_ALREADY_OPEN	See “Cursor already open” [<i>SQL Anywhere 10 - Error Messages</i>].	-172
SQLE_CURSOR_NOT_DECLARED	See “Cursor has not been declared” [<i>SQL Anywhere 10 - Error Messages</i>].	-170

Member name	Description	Value
SQLE_CURSOR_NOT_OPEN	See “Cursor not open” [SQL Anywhere 10 - Error Messages].	-180
SQLE_CURSOR_RESTORED	See “UltraLite cursor (or result set or table) was restored” [SQL Anywhere 10 - Error Messages].	134
SQLE_CURSOROP_NOT_ALLOWED	See “Illegal cursor operation attempt” [SQL Anywhere 10 - Error Messages].	-187
SQLE_DATABASE_ERROR	See “Internal database error %1 -- transaction rolled back” [SQL Anywhere 10 - Error Messages].	-301
SQLE_DATABASE_NAME_REQUIRED	See “Database name required to start server” [SQL Anywhere 10 - Error Messages].	-87
SQLE_DATABASE_NOT_CREATED	See “Database creation failed: %1” [SQL Anywhere 10 - Error Messages].	-645
SQLE_DATATYPE_NOT_ALLOWED	See “Expression has unsupported data type” [SQL Anywhere 10 - Error Messages].	-624
SQLE_DBSPACE_FULL	See “A dbspace has reached its maximum file size” [SQL Anywhere 10 - Error Messages].	-604
SQLE_DESCRIBE_NONSELECT	See “Can only describe a SELECT statement” [SQL Anywhere 10 - Error Messages].	-160
SQLE_DEVICE_ERROR	See “I/O error %1 -- transaction rolled back” [SQL Anywhere 10 - Error Messages].	-305
SQLE_DEVICE_IO_FAILED	See “File I/O failed for '%1'” [SQL Anywhere 10 - Error Messages].	-974
SQLE_DIV_ZERO_ERROR	See “Division by zero” [SQL Anywhere 10 - Error Messages].	-628
SQLE_DOWNLOAD_CONFLICT	See “Download failed because of conflicts with existing rows” [SQL Anywhere 10 - Error Messages].	-839
SQLE_DOWNLOAD_RESTART_FAILED	See “Unable to retry download because upload is not finished” [SQL Anywhere 10 - Error Messages].	-1102
SQLE_DROP_DATABASE_FAILED	See “An attempt to delete database '%1' failed” [SQL Anywhere 10 - Error Messages].	-651
SQLE_DUPLICATE_CURSOR_NAME	See “The cursor name '%1' already exists” [SQL Anywhere 10 - Error Messages].	-683
SQLE_DUPLICATE_FOREIGN_KEY	See “Foreign key '%1' for table '%2' duplicates an existing foreign key” [SQL Anywhere 10 - Error Messages].	-251

Member name	Description	Value
SQLC_DUPLICATE_OPTION	See “Option '%1' specified more than once” [SQL Anywhere 10 - Error Messages].	139
SQLC_DYNAMIC_MEMORY_EXHAUSTED	See “Dynamic memory exhausted” [SQL Anywhere 10 - Error Messages].	-78
SQLC_ENCRYPTION_INITIALIZATION_FAILED	See “Could not initialize the encryption DLL: '%1'” [SQL Anywhere 10 - Error Messages].	-984
SQLC_ENCRYPTION_NOT_ENABLED	See “Encryption has not been enabled” [SQL Anywhere 10 - Error Messages].	-1143
SQLC_ENCRYPTION_NOT_ENABLED_WARNING	See “Encryption has not been enabled” [SQL Anywhere 10 - Error Messages].	140
SQLC_ENGINE_ALREADY_RUNNING	See “Database server already running” [SQL Anywhere 10 - Error Messages].	-96
SQLC_ERROR	See “Run time SQL error -- %1” [SQL Anywhere 10 - Error Messages].	-300
SQLC_ERROR_CALLING_FUNCTION	See “Could not allocate resources to call external function” [SQL Anywhere 10 - Error Messages].	-622
SQLC_ERROR_IN_ASSIGNMENT	See “Error in assignment” [SQL Anywhere 10 - Error Messages].	-641
SQLC_EXPRESSION_ERROR	See “Invalid expression near '%1'” [SQL Anywhere 10 - Error Messages].	-156
SQLC_FEATURE_NOT_ENABLED	See “The method you attempted to invoke was not enabled for your application” [SQL Anywhere 10 - Error Messages].	-1092
SQLC_FILE_BAD_DB	See “Unable to start specified database: '%1' is not a valid database file” [SQL Anywhere 10 - Error Messages].	-1006
SQLC_FILE_IN_USE	See “Specified database file already in use” [SQL Anywhere 10 - Error Messages].	-816
SQLC_FILE_NOT_DB	See “Unable to start specified database: '%1' is not a database” [SQL Anywhere 10 - Error Messages].	-1004
SQLC_FILE_VOLUME_NOT_FOUND	See “Specified file system volume not found for database '%1'” [SQL Anywhere 10 - Error Messages].	-1112

Member name	Description	Value
SQLE_FILE_WRONG_VERSION	See “Unable to start specified database: '%1' was created by a different version of the software” [SQL Anywhere 10 - Error Messages].	-1005
SQLE_FOREIGN_KEY_NAME_NOT_FOUND	See “Foreign key name '%1' not found” [SQL Anywhere 10 - Error Messages].	-145
SQLE_IDENTIFIER_TOO_LONG	See “Identifier '%1' too long” [SQL Anywhere 10 - Error Messages].	-250
SQLE_INCORRECT_VOLUME_ID	See “Incorrect volume ID for '%1'” [SQL Anywhere 10 - Error Messages].	-975
SQLE_INDEX_NAME_NOT_UNIQUE	See “Index name '%1' not unique” [SQL Anywhere 10 - Error Messages].	-111
SQLE_INDEX_NOT_FOUND	See “Cannot find index named '%1'” [SQL Anywhere 10 - Error Messages].	-183
SQLE_INDEX_NOT_UNIQUE	See “Index '%1' for table '%2' would not be unique” [SQL Anywhere 10 - Error Messages].	-196
SQLE_INTERRUPTED	See “Statement interrupted by user” [SQL Anywhere 10 - Error Messages].	-299
SQLE_INVALID_CONSTRAINT_REF	See “Invalid reference to or operation on constraint '%1'” [SQL Anywhere 10 - Error Messages].	-937
SQLE_INVALID_DESCRIPTOR_INDEX	See “Invalid descriptor index” [SQL Anywhere 10 - Error Messages].	-640
SQLE_INVALID_DESCRIPTOR_NAME	See “Invalid SQL descriptor name” [SQL Anywhere 10 - Error Messages].	-642
SQLE_INVALID_DISTINCT_AGGREGATE	See “Grouped query contains more than one distinct aggregate function” [SQL Anywhere 10 - Error Messages].	-863
SQLE_INVALID_FOREIGN_KEY	See “No primary key value for foreign key '%1' in table '%2'” [SQL Anywhere 10 - Error Messages].	-194
SQLE_INVALID_FOREIGN_KEY_DEF	See “Column '%1' in foreign key has a different definition than primary key” [SQL Anywhere 10 - Error Messages].	-113
SQLE_INVALID_GROUP_SELECT	See “Function or column reference to '%1' must also appear in a GROUP BY” [SQL Anywhere 10 - Error Messages].	-149
SQLE_INVALID_INDEX_TYPE	See “Index type specification of '%1' is invalid” [SQL Anywhere 10 - Error Messages].	-650

Member name	Description	Value
SQLE_INVALID_LOGON	See “Invalid user ID or password” [SQL Anywhere 10 - Error Messages].	-103
SQLE_INVALID_OPTION	See “Invalid option '%1' -- no PUBLIC setting exists” [SQL Anywhere 10 - Error Messages].	-200
SQLE_INVALID_OPTION_SETTING	See “Invalid setting for option '%1'” [SQL Anywhere 10 - Error Messages].	-201
SQLE_INVALID_OPTION_VALUE	See “‘%1’ is an invalid value for ‘%2’” [SQL Anywhere 10 - Error Messages].	-1053
SQLE_INVALID_ORDER	See “Invalid ORDER BY specification” [SQL Anywhere 10 - Error Messages].	-152
SQLE_INVALID_PARAMETER	See “Invalid parameter” [SQL Anywhere 10 - Error Messages].	-735
SQLE_INVALID_PARSE_PARAMETER	See “Parse error: %1” [SQL Anywhere 10 - Error Messages].	-95
SQLE_INVALID_PUBLICATION_MASK	See “The specified publication mask is invalid” [SQL Anywhere 10 - Error Messages].	-1105
SQLE_INVALID_SQL_IDENTIFIER	See “Invalid SQL identifier” [SQL Anywhere 10 - Error Messages].	-760
SQLE_INVALID_STATEMENT	See “Invalid statement” [SQL Anywhere 10 - Error Messages].	-130
SQLE_INVALID_UNION	See “Select lists in UNION, INTERSECT, or EXCEPT do not match in length” [SQL Anywhere 10 - Error Messages].	-153
SQLE_KEYLESS_ENCRYPTION	See “Unable to perform requested operation since this database uses keyless encryption” [SQL Anywhere 10 - Error Messages].	-1109
SQLE_LOCKED	See “User '%1' has the row in '%2' locked” [SQL Anywhere 10 - Error Messages].	-210
SQLE_MAX_ROW_SIZE_EXCEEDED	See “Maximum row size for table '%1' would be exceeded” [SQL Anywhere 10 - Error Messages].	-1132
SQLE_MEMORY_ERROR	See “Memory error -- transaction rolled back” [SQL Anywhere 10 - Error Messages].	-309
SQLE_METHOD_CANNOT_BE_CALLED	See “Method '%1' cannot be called at this time” [SQL Anywhere 10 - Error Messages].	-669
SQLE_MIRROR_FILE_MISMATCH	See “The mirror '%1' does not match database '%2’” [SQL Anywhere 10 - Error Messages].	-1138

Member name	Description	Value
SQLE_NAME_NOT_UNIQUE	See “Item '%1' already exists” [SQL Anywhere 10 - Error Messages].	-110
SQLE_NO_COLUMN_NAME	See “Derived table '%1' has no name for column %2” [SQL Anywhere 10 - Error Messages].	-163
SQLE_NO_CURRENT_ROW	See “No current row of cursor” [SQL Anywhere 10 - Error Messages].	-197
SQLE_NO_INDICATOR	See “No indicator variable provided for NULL result” [SQL Anywhere 10 - Error Messages].	-181
SQLE_NO_MATCHING_SELECT_ITEM	See “The select list for the derived table '%1' has no expression to match '%2'” [SQL Anywhere 10 - Error Messages].	-812
SQLE_NO_PRIMARY_KEY	See “Table '%1' has no primary key” [SQL Anywhere 10 - Error Messages].	-118
SQLE_NOERROR	SQLE_NOERROR(0) - This code indicates that there was no error or warning.	0
SQLE_NON_UPDATABLE_COLUMN	See “Cannot update an expression” [SQL Anywhere 10 - Error Messages].	-190
SQLE_NON_UPDATABLE_CURSOR	See “FOR UPDATE has been incorrectly specified for a READ ONLY cursor” [SQL Anywhere 10 - Error Messages].	-813
SQLE_NOT_IMPLEMENTED	See “Feature '%1' not implemented” [SQL Anywhere 10 - Error Messages].	-134
SQLE_NOT_SUPPORTED_IN_ULTRALITE	See “Feature not available with UltraLite” [SQL Anywhere 10 - Error Messages].	-749
SQLE_NOTFOUND	See “Row not found” [SQL Anywhere 10 - Error Messages].	100
SQLE_ONLY_ONE_TABLE	See “INSERT/DELETE on cursor can modify only one table” [SQL Anywhere 10 - Error Messages].	-199
SQLE_OVERFLOW_ERROR	See “Value %1 out of range for destination” [SQL Anywhere 10 - Error Messages].	-158
SQLE_PAGE_SIZE_INVALID	See “Invalid database page size” [SQL Anywhere 10 - Error Messages].	-644
SQLE_PARTIAL_DOWNLOAD_NOT_FOUND	See “No partial download was found” [SQL Anywhere 10 - Error Messages].	-1103
SQLE_PERMISSION_DENIED	See “Permission denied: %1” [SQL Anywhere 10 - Error Messages].	-121

Member name	Description	Value
SQLC_PRIMARY_KEY_NOT_UNIQUE	See “Primary key for table '%1' is not unique” [SQL Anywhere 10 - Error Messages].	-193
SQLC_PRIMARY_KEY_TWICE	See “Table cannot have two primary keys” [SQL Anywhere 10 - Error Messages].	-126
SQLC_PRIMARY_KEY_VALUE_REF	See “Primary key for row in table '%1' is referenced by foreign key '%2' in table '%3'” [SQL Anywhere 10 - Error Messages].	-198
SQLC_PUBLICATION_NOT_FOUND	See “Publication '%1' not found” [SQL Anywhere 10 - Error Messages].	-280
SQLC_PUBLICATION_PREDICATE_IGNORED	See “Publication predicates were not evaluated” [SQL Anywhere 10 - Error Messages].	138
SQLC_RESOURCE_GOVERNOR_EXCEEDED	See “Resource governor for '%1' exceeded” [SQL Anywhere 10 - Error Messages].	-685
SQLC_ROW_DELETED_TO_MAINTAIN_REFERENTIAL_INTEGRITY	See “Row was dropped from table %1 to maintain referential integrity” [SQL Anywhere 10 - Error Messages].	137
SQLC_ROW_EXCEEDS_PAGE_SIZE	See “A row cannot be stored because it exceeds the database page size” [SQL Anywhere 10 - Error Messages].	-1117
SQLC_SCHEMA_UPGRADE_NOT_ALLOWED	See “A schema upgrade is not currently allowed” [SQL Anywhere 10 - Error Messages].	-953
SQLC_SERVER_SYNCHRONIZATION_ERROR	See “Synchronization failed due to an error on the server: %1” [SQL Anywhere 10 - Error Messages].	-857
SQLC_START_STOP_DATABASE_DENIED	See “Request to start/stop database denied” [SQL Anywhere 10 - Error Messages].	-75
SQLC_STATEMENT_ERROR	See “SQL statement error” [SQL Anywhere 10 - Error Messages].	-132
SQLC_STRING_RIGHT_TRUNCATION	See “Right truncation of string data” [SQL Anywhere 10 - Error Messages].	-638
SQLC_SUBQUERY_SELECT_LIST	See “Subquery allowed only one select list item” [SQL Anywhere 10 - Error Messages].	-151

Member name	Description	Value
SQLE_SYNC_INFO_INVALID	See “Information for synchronization is incomplete or invalid, check '%1'” [<i>SQL Anywhere 10 - Error Messages</i>].	-956
SQLE_SYNC_INFO_REQUIRED	See “Information for synchronization was not provided” [<i>SQL Anywhere 10 - Error Messages</i>].	-1111
SQLE_SYNC_NOT_REENTRANT	See “Synchronization process was unable to re-enter synchronization” [<i>SQL Anywhere 10 - Error Messages</i>].	-1110
SQLE_SYNC_STATUS_UNKNOWN	See “The status of the last synchronization upload is unknown” [<i>SQL Anywhere 10 - Error Messages</i>].	-952
SQLE_SYNTAX_ERROR	See “Syntax error near '%1' %2” [<i>SQL Anywhere 10 - Error Messages</i>].	-131
SQLE_TABLE_ALREADY_INCLUDED	See “Table '%1' is already included” [<i>SQL Anywhere 10 - Error Messages</i>].	-822
SQLE_TABLE_IN_USE	See “Table in use” [<i>SQL Anywhere 10 - Error Messages</i>].	-214
SQLE_TABLE_NOT_FOUND	See “Table '%1' not found” [<i>SQL Anywhere 10 - Error Messages</i>].	-141
SQLE_TOO_MANY_BLOB_REFS	See “Too many references to a BLOB” [<i>SQL Anywhere 10 - Error Messages</i>].	-1107
SQLE_TOO_MANY_CONNECTIONS	See “Database server connection limit exceeded” [<i>SQL Anywhere 10 - Error Messages</i>].	-102
SQLE_TOO_MANY_PUBLICATIONS	See “Too many publications specified in publication mask” [<i>SQL Anywhere 10 - Error Messages</i>].	-1106
SQLE_TOO_MANY_TEMP_TABLES	See “Too many temporary tables in connection” [<i>SQL Anywhere 10 - Error Messages</i>].	-817
SQLE_TOO_MANY_USERS	See “Too many users in database” [<i>SQL Anywhere 10 - Error Messages</i>].	-1104
SQLE_ULTRALITE_DATABASE_NOT_FOUND	See “The database '%1' was not found” [<i>SQL Anywhere 10 - Error Messages</i>].	-954
SQLE_ULTRALITE_OBJ_CLOSED	See “Invalid operation on a closed object” [<i>SQL Anywhere 10 - Error Messages</i>].	-908
SQLE_ULTRALITE_WRITE_ACCESS_DENIED	See “Write access was denied” [<i>SQL Anywhere 10 - Error Messages</i>].	-1086

Member name	Description	Value
SQLE_UNABLE_TO_CONNECT	See “Database cannot be started -- %1” [<i>SQL Anywhere 10 - Error Messages</i>].	-105
SQLE_UNABLE_TO_START_DATABASE	See “Unable to start specified database: %1” [<i>SQL Anywhere 10 - Error Messages</i>].	-82
SQLE_UNABLE_TO_START_DATABASE_VER_NEWER	See “Unable to start specified database: Server must be upgraded to start database %1” [<i>SQL Anywhere 10 - Error Messages</i>].	-934
SQLE_UNCOMMITTED_TRANSACTIONS	See “You cannot synchronize or upgrade with uncommitted transactions” [<i>SQL Anywhere 10 - Error Messages</i>].	-755
SQLE_UNKNOWN_FUNCTION	See “Unknown function '%1'” [<i>SQL Anywhere 10 - Error Messages</i>].	-148
SQLE_UNKNOWN_OPTION	See “'%1' is an unknown option” [<i>SQL Anywhere 10 - Error Messages</i>].	120
SQLE_UNKNOWN_USERID	See “User ID '%1' does not exist” [<i>SQL Anywhere 10 - Error Messages</i>].	-140
SQLE_UNRECOGNIZED_OPTION	See “The option '%1' is not recognized” [<i>SQL Anywhere 10 - Error Messages</i>].	-1002
SQLE_UPLOAD_FAILED_AT_SERVER	See “Synchronization server failed to commit the upload” [<i>SQL Anywhere 10 - Error Messages</i>].	-794
SQLE_VALUE_IS_NULL	See “Cannot return NULL result as requested data type” [<i>SQL Anywhere 10 - Error Messages</i>].	-1050
SQLE_VARIABLE_INVALID	See “Invalid host variable” [<i>SQL Anywhere 10 - Error Messages</i>].	-155
SQLE_WRONG_NUMBER_OF_INSERT_COLUMNS	See “Wrong number of values for INSERT” [<i>SQL Anywhere 10 - Error Messages</i>].	-207
SQLE_WRONG_PARAMETER_COUNT	See “Wrong number of parameters to function '%1'” [<i>SQL Anywhere 10 - Error Messages</i>].	-154

ULStreamErrorCode enumeration

UL Ext.: Enumerates the error codes that may be reported by streams during synchronization.

Syntax

Visual Basic

Public Enum **ULStreamErrorCode**

C#

public enum **ULStreamErrorCode**

Members

Member name	Description	Value
ACTSYNC_NO_PORT	ActiveSync synchronization can only be initiated by ActiveSync itself, either by placing the device in its cradle or by selecting "Synchronize" from the ActiveSync Manager. To initiate a synchronization from an application, use the TCP/IP socket synchronization stream.	75
ACTSYNC_NOT_INSTALLED	The ActiveSync provider has not been installed. Run dbasinst to install it (see documentation for details).	76
CONNECT_TIMEOUT	The connection attempt timed out. Either the server is not running on the indicated host and port or the timeout value needs to be increased to allow more time to connect.	241
COULD_NOT_OPEN_FILE_FOR_WRITE	The specified file could not be opened for write. Make sure that this is the correct file and that no other application is using it.	230
CREATE_RANDOM_OBJECT	The secure network layer could not create a random-number-generating object. Free up system resources, reconnect and retry the operation.	17
DEQUEUEING_CONNECTION	The MobiLink server encountered an error while attempting to get a queued connection (synchronization) request. Free up system resources. If the problem persists, restart the MobiLink server.	19
DUN_DIAL_FAILED	Automatic dialup failed to establish connection to the specified dial up network.	204
DUN_NOT_SUPPORTED	An attempt to dialup has failed due to insufficient system support. On PocketPC you must use cellcore.dll and on Windows you must use wininet.dll from IE 4.0 or above. Dialup is not supported on other platforms.	203

Member name	Description	Value
END_READ	Unable to finish a sequence of reads from the network. See also: READ	11
END_WRITE	Unable to finish a sequence of writes to the network. See also: WRITE	10
GENERATE_RANDOM	The secure network layer requires a random number but was unable to generate one. Free up system resources, reconnect and retry the operation.	14
HTTP_AUTHENTICATION_FAILED	The supplied userid and password were rejected. Check that they were entered correctly. If so, contact your systems administrator to ensure you have proper access.	211
HTTP_AUTHENTICATION_REQUIRED	An HTTP server or gateway requested HTTP authentication. Please supply a userid and password using the HTTP synchronization parameters http_userid and http_password.	209
HTTP_BAD_STATUS_CODE	Examine the status line to determine the cause of the failure.	86
HTTP_BUFFER_SIZE_OUT_OF_RANGE	Fix the HTTP buffer size. A valid buffer size is positive and not overly large for the host platform.	79
HTTP_CHUNK_LEN_BAD_CHARACTER	Try using a fixed length HTTP body.	85
HTTP_CHUNK_LEN_ENCODED_MISSING	Try using a fixed length HTTP body.	84
HTTP_CLIENT_ID_NOT_SET	The client id was not passed into the HTTP client code. Contact technical support for a fix.	78
HTTP_CONTENT_TYPE_NOT_SPECIFIED	An unknown content type was specified. Refer to the documentation and change the content type to one of the supported types.	77
HTTP_CRLF_ENCODED_MISSING	The proxy you are using may not be compatible with MobiLink. Please check your configuration.	81
HTTP_CRLF_MISSING	The proxy you are using may not be compatible with MobiLink. Please check your configuration.	82
HTTP_EXPECTED_POST	The proxy you are using may not be compatible with MobiLink. Please check your configuration.	89
HTTP_EXTRA_DATA_END_READ	Extra data has been introduced into the HTTP body. This may have been added by a proxy agent. Try eliminating the proxy.	80

Member name	Description	Value
HTTP_HEADER_PARSE_ERROR	An error occurred while trying to parse an HTTP header. The header may be malformed.	216
HTTP_INTERNAL_HEADER_STATE	There was a problem decoding the HTTP header. This is an internal error that should never occur. Please contact technical support.	236
HTTP_INTERNAL_REQUEST_TYPE	There was a problem determining the HTTP request type. This is an internal error that should never occur. Please contact technical support.	237
HTTP_INVALID_CHARACTER	An unexpected character was read in an HTTP header. The header may be malformed or the other side may not be sending HTTP at all.	219
HTTP_INVALID_SESSION_KEY	An unknown session key type was specified. Refer to the documentation and change the session key type to one of the supported types.	244
HTTP_MALFORMED_SESSION_COOKIE	The HTTP cookie used to manage the synchronization session is corrupt. Determine where the cookie is being corrupted. The most likely cause is a client error, or perhaps an HTTP intermediary misbehaving.	235
HTTP_NO_CONNECTION	The server timed out while waiting for the next HTTP request from the remote site. Determine why this request failed to reach the server or try a persistent connection.	83
HTTP_NO_PASSWORD	A userid was supplied for HTTP authentication but no password. Both are required for authentication.	214
HTTP_NO_USERID	A password was supplied for HTTP authentication but no userid. Both are required for authentication.	213
HTTP_PROXY_AUTHENTICATION_FAILED	The supplied userid and password were rejected by the proxy server. Check that they were entered correctly. If so, contact your systems administrator to ensure you have proper access.	212
HTTP_PROXY_AUTHENTICATION_REQUIRED	An HTTP proxy requested HTTP authentication. Please supply a userid and password using the HTTP synchronization parameters <code>http_proxy_userid</code> and <code>http_proxy_password</code> .	210
HTTP_SERVER_AUTH_FAILED	The Authentication-Info header sent from the server contained an incorrect value, causing authentication to fail. Make sure that you are connecting to a legitimate HTTP server.	217
HTTP_UNABLE_TO_PARSE_COOKIE	Determine where the set cookie header is being corrupted.	88

Member name	Description	Value
HTTP_UNKNOWN_TRANSFER_ENCODING	Determine how the unknown transfer encoding is getting generated.	87
HTTP_UNSUPPORTED_AUTH_ALGORITHM	The HTTP Digest authentication algorithm requested by the server is unsupported. Only "MD5" and "MD5-sess" are supported.	215
HTTP_VERSION	The requested HTTP version is unsupported. Consult the documentation and specify a supported HTTP version. At the time of publication the supported HTTP versions are 1.0 and 1.1.	54
INCONSISTENT_FIPS	Use of the -fips switch on the MobiLink server command line requires that all secure streams be FIPS-compliant. If a secure stream is not configured with the fips option, it will automatically be FIPS-compliant (for example, fips=y). Either remove the fips option from the secure stream, or enable it with fips=y.	242
INIT_RANDOM	The secure network layer could not initialize its random number generator. Free up system resources, reconnect and retry the operation.	15
INTERNAL	An internal error has occurred in the network layer. Please contact technical support.	220
INTERNAL_API	An internal error has occurred in the network layer. Please contact technical support.	238
INTERNAL_PROTOCOL_NOT_LOADED	A synchronization protocol could not be loaded. If you are using UltraLite, make sure you have called the proper ULEnable method.	227
INTERRUPTED	The current operation was interrupted by the caller.	218
INVALID_COMPRESSION_TYPE	The specified compression type was not recognized.	232
INVALID_LOCAL_PATH	The local path for the downloaded file is invalid. Consult the documentation for details.	243
INVALID_NETWORK_LIBRARY	The given network interface DLL or shared object could not be loaded, possibly because it is invalid or corrupt.	245
INVALID_SYNC_PROTOCOL	The specified protocol is not a valid synchronization protocol.	226
LIBRARY_ENTRY_POINT_NOT_FOUND	The indicated library entry point could not be found.	225

Member name	Description	Value
LOAD_LIBRARY_FAILURE	The indicated library could not be found in the path. If you are trying to use TLS encryption for synchronization, make sure you have acquired the proper license.	224
LOAD_NETWORK_LIBRARY	The network interface library could not be found and/or loaded. Please check the following: 1) The sockets layer is properly installed. The correct network interface library (or DLL or shared object) must be present and accessible. 2) There are enough system resources available. Free up system resources if they are running low.	73
MEMORY_ALLOCATION	The network layer was unable to allocate enough bytes of storage. Free up system memory and retry the operation. The technique used to free up system memory depends on the operating system and how it is configured. The simplest technique is to reduce the number of active processes. Consult your operating system documentation for details.	6
MISSING_PARAMETER	The specified parameter was expected but not supplied.	229
NETWORK_LIBRARY_VERSION_MISMATCH	A network interface DLL or shared object could not be loaded because it is the wrong version.	248
NO_ECC_FIPS	There was a problem performing the given compression operation. Please contact technical support.	239
NONE	This code indicates there was either no network error, or an unknown network error occurred.	0
NOT_IMPLEMENTED	An unimplemented internal feature was requested. Please contact technical support.	12
PARAMETER	Network parameters are of the form "name=value; [name2=value2;...]". This code indicates an invalid parameter value. Consult the documentation for the corresponding parameter name, and correct the parameter value.	1
PARAMETER_NOT_BOOLEAN	Network parameters are of the form "name=value; [name2=value2;...]". The parameter value is not a boolean value. Locate the offending parameter specification and change the value of the parameter to either 0 (for off or false) or 1 (for on or true).	4

Member name	Description	Value
PARAMETER_NOT_HEX	Network parameters are of the form "name=value; [name2=value2[;...]]". The parameter value is not a hexadecimal (base 16) value. Locate the offending parameter specification and change the value of the parameter to a hexadecimal value.	5
PARAMETER_NOT_UINT32	Network parameters are of the form "name=value; [name2=value2[;...]]". The parameter value is not an unsigned integer. Locate the offending parameter specification and change the value of the parameter to an unsigned integer.	2
PARAMETER_NOT_UINT32_RANGE	Network parameters are of the form "name=value; [name2=value2[;...]]". The parameter value is not an unsigned integer value or range. Locate the offending parameter specification and change the value of the parameter to an unsigned integer or an unsigned range. An unsigned range has the form: NNN-NNN.	3
PARSE	Network parameters are of the form "name=value; [name2=value2[;...]]". Optionally, the entire list of parameters may be enclosed in parentheses. The given string does not follow this convention. Inspect the string, fix any formatting problems, and retry the operation.	7
PROTOCOL_ERROR	An unexpected value or token was read.	231

Member name	Description	Value
READ	Unable to read the given number of bytes from the network layer. Note that reads may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part of a basic operation in the upper layer. The cause of a read error is usually one of the following: 1) The network had a problem that caused the read to fail. Reconnect and retry the operation. 2) The connection timed out. Reconnect and retry the operation. 3) The other side of the connection cleanly terminated the connection. Consult the client and/or server logs for errors that indicate why the connection has been dropped. Consult the output-log errors and fix the cause, then retry the operation. 4) The process at the other side of the connection was aborted. Consult the client and/or server output logs for errors that indicate why the process was aborted. If the process was shut down by other than normal means, there may not be any errors in its output log. Reconnect and retry the operation. 5) The system is low on resources, and cannot perform the read. Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.	8
READ_TIMEOUT	Unable to read the given number of bytes from the network layer in the given time. Check that the network is functioning correctly, and that the sending application is still running.	201
SACI_ERROR_CLIENT	An error has been reported by the SACI implementation.	246
SACI_ERROR_SERVER	The SACI network library is reporting an error. Refer to the provider of the SACI network library to resolve the problem.	247
SECURE_ADD_CERTIFICATE	The secure network layer was unable to add a certificate to a certificate chain. Free up system resources and retry the operation.	39
SECURE_ADD_TRUSTED_CERTIFICATE	The secure network layer was unable to add a trusted certificate to a certificate chain. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.	48

Member name	Description	Value
SECURE_CERTIFICATE_COMMON_NAME	The given common name is not in the certificate chain. Check the following: 1) The common name was properly entered. 2) The correct certificate file was specified. 3) The common name is in the certificate chain. You can verify this with the readcert utility.	52
SECURE_CERTIFICATE_COMPANY_NAME	The given organization name is not in the certificate chain. Check the following: 1) The organization name was properly entered. 2) The correct certificate file was specified. 3) The organization name is in the certificate chain. You can verify this with the readcert utility.	21
SECURE_CERTIFICATE_COMPANY_UNIT	The given organization unit is not in the certificate chain. Check the following: 1) The in company name was properly entered. 2) The correct certificate file was specified. 3) The company name is in the certificate chain. You can verify this with the readcert utility.	51
SECURE_CERTIFICATE_COUNT	The given file does not contain a certificate. Check the following: 1) The certificate file name was properly specified. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s).	42
SECURE_CERTIFICATE_EXPIRED	A certificate in the certificate chain has expired. Obtain a new certificate with a later expiry date and retry the operation.	50
SECURE_CERTIFICATE_EXPIRY_DATE	A certificate's expiry date could not be read. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.	37

Member name	Description	Value
SECURE_CERTIFICATE_FILE_NOT_FOUND	The certificate file could not be opened. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to read the file. This only applies to operating systems having user and/or file permissions.	33
SECURE_CERTIFICATE_NOT_TRUSTED	The server's certificate was not signed by a trusted authority. Check the following: 1) The certificate file name was properly specified. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The client's list of trusted root certificates includes the server's root certificate.	24
SECURE_CERTIFICATE_ROOT	The root certificate in the chain is invalid. At the time of publication, this error was defined but not used.	20
SECURE_CREATE_CERTIFICATE	The secure network layer was unable to allocate storage for a certificate. Free up system resources and retry the operation.	43
SECURE_CREATE_PRIVATE_KEY_OBJECT	The secure network layer was unable to create a private key object, prior to loading the private key. The most likely cause is a shortage of system resources. Free up system resources and retry the operation.	49
SECURE_DUPLICATE_CONTEXT	The secure network layer was unable to duplicate a security context. Free up system resources and retry the operation.	25
SECURE_EXPORT_CERTIFICATE	The secure network layer was unable to copy a certificate. Free up system resources and retry the operation.	38

Member name	Description	Value
SECURE_HANDSHAKE	The secure handshake failed. Check the following: 1) On the client, the correct host machine and port number were specified. 2) On the server, the correct port number was specified. 3) The correct certificate file was specified, both on the client and on the server.	53
SECURE_IMPORT_CERT_FROM_SYSTEM_STORE	Failed to import a certificate from the system certificate store.	222
SECURE_IMPORT_CERTIFICATE	The secure network layer was unable to import a certificate. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s).	44
SECURE_NO_CERTS_IN_SYS_STORE	No certificates were found in the system's certificate store.	223
SECURE_NO_SERVER_CERTIFICATE	No server certificate was provided. A server certificate is required for secure communications. The file provided must contain the full chain of certificates for the server as well as its private key.	205
SECURE_NO_SERVER_CERTIFICATE_PASSWORD	No server certificate password was provided. This password is required to decrypt the server's encrypted private key.	206
SECURE_NO_TRUSTED_ROOTS	No trusted root certificates were provided. At least one trusted root certificate is required for secure communications.	207
SECURE_OPEN_SYSTEM_CERT_STORE	An attempt to open a system certificate store failed.	221

Member name	Description	Value
SECURE_READ_CERTIFICATE	The certificate file could not be read. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.	34
SECURE_READ_PRIVATE_KEY	The private key could not be read from the certificate file. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.	35
SECURE_REDUNDANT_SERVER_CERTIFICATE_PASSWORD	A password was specified when the server's private key wasn't encrypted by any password.	208
SECURE_SET_IO	The secure network layer was unable to attach to the network layer. Free up system resources and retry the operation.	26
SECURE_SET_PRIVATE_KEY	The private key could not be used. Check the following: 1) The password was entered correctly. 2) The certificate file contains one or more certificates. 3) The certificate file contains the correct certificate(s). 4) The certificate file is undamaged.	36

Member name	Description	Value
SECURE_TRUSTED_CERTIFICATE_FILE_NOT_FOUND	The certificate file could not be found. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.	40
SECURE_TRUSTED_CERTIFICATE_READ	The secure network layer was unable to read the trusted certificate file. Check the following: 1) The certificate file name was properly specified. 2) The certificate file exists. 3) The certificate file contains one or more certificates. 4) The certificate file contains the correct certificate(s). 5) The program attempting to open the certificate file has sufficient privileges to see the file. This only applies to operating systems having user and/or file permissions.	41
SEED_RANDOM	The secure network layer could not seed its random number generator. Free up system resources, re-connect and retry the operation.	16
SERVER_ERROR	The server reported an error. Contact the MobiLink administrator to learn more.	228
SHUTTING_DOWN	The MobiLink server encountered an error in the network layer during shutdown. It is possible that some network operations pending at the time of shutdown were affected.	18

Member name	Description	Value
SOCKET_BIND	The network layer was unable to bind a socket to the given port. Check the following. 1) (Server only) Verify that the port isn't already in use. If the port is in use, either shut down the application listening on that port, or specify a different port. 2) (Server only) Verify that there are no firewall restrictions on the use of the port. 3) (Client only) If the client_port option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (for example, NNN-NNN). If a range was specified, consider making it a wider range, or a different range. 4) (Client only) If the client_port option was used, verify that there are no firewall restrictions on the use of the port.	60
SOCKET_CLEANUP	The network layer was unable to clean up the socket layer. This error should only occur after all connections are finished, so no current connections should be affected.	61
SOCKET_CLOSE	The network layer was unable to close a socket. The network session may or may not have terminated prematurely, due to pending writes that were not flushed. Check the following: 1) The other side of the network connection had any errors. 2) The other side of the connection is running normally. 3) The machine is still connected to the network, and the network is responsive.	62

Member name	Description	Value
SOCKET_CONNECT	The network layer was unable to connect a socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 3) The correct host machine and port were specified. 4) The host server is running normally and listening on the correct port. 5) The host machine is listening for the proper socket type (TCP/IP vs. UDP). 6) If the client_port option was used, verify that there are no firewall restrictions on the use of the port. 7) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 8) There are enough system resources available. Free up system resources if they are running low.	63
SOCKET_CREATE_TCPIP	The network layer was unable to create a TCP/IP socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 6) There are enough system resources available. Free up system resources if they are running low.	58

Member name	Description	Value
SOCKET_CREATE_UDP	The network layer was unable to create a UDP socket. Check the following: 1) The machine is connected to the network. 2) The socket layer is properly initialized. 3) If the client_port option was used, verify that the given port isn't already in use. If only one client port was specified, consider using a range (for example, NNN-NNN). If a range was specified, consider making it a wider range, or a different range. 4) If the client_port option was used, verify that there are no firewall restrictions on the use of the port. 5) If the device has a limit on the number of open sockets, verify that the limit has not been reached. 6) There are enough system resources available. Free up system resources if they are running low.	59
SOCKET_GET_HOST_BY_ADDR	The network layer was unable to get the name of a host using its IP address. At the time of publication, this error was defined but not used.	72
SOCKET_GET_NAME	The network layer was unable to determine a socket's local name. In a TCP/IP connection, each end of the connection has a socket exclusively attached to a port. A socket's local name includes this port number, which is assigned by the network at connection time. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.	64
SOCKET_GET_OPTION	The network layer was unable to get a socket option. This error may be the first indication that a connection has been lost. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.	65

Member name	Description	Value
SOCKET_HOST_NAME_NOT_FOUND	The given host name could not be found. Check the following: 1) The host name was correctly specified. 2) The host is accessible. Many systems include a "ping" utility that can be used to verify access to a named host. 3) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try specifying the host's IP number (for example, NNN.NNN.NNN.NNN) instead of the host name. 4) The HOSTS file contains an entry that maps the host name to an IP number.	57
SOCKET_LISTEN	The server is unable to listen on a socket. The backlog refers to the maximum number of queued connection requests that may be pending at any given time. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) There are no firewall or other restrictions preventing a socket listener from running on the current machine. 3) The backlog setting is within the limit, if any, on the machine. 4) There are enough system resources available. Free up system resources if they are running low.	67
SOCKET_LIVENESS_OUT_OF_RANGE	An invalid liveness timeout value was specified. The liveness timeout value must be an integer between zero and 65535.	200
SOCKET_LOCALHOST_NAME_NOT_FOUND	The network layer was unable to determine the IP address of "localhost". Check the following: 1) The Domain Name Server (DNS), or its equivalent, is available. If the DNS is not available, try explicitly specifying the localhost IP number (usually 127.0.0.1) instead. 2) The HOSTS file contains an entry that maps the "localhost" name to an IP number. 3) There are enough system resources available. Free up system resources if they are running low.	71
SOCKET_PORT_OUT_OF_RANGE	An invalid port number was specified. The port number must be an integer between zero and 65535.	74

Member name	Description	Value
SOCKET_SELECT	The network layer encountered an error attempting to wait for a socket to be ready for reading or writing. Check the following: 1) The machine is connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.	69
SOCKET_SET_OPTION	The network layer was unable to set a socket option. This error may be the first indication that a connection has been lost. Check the following: 1) The machine is still connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.	66
SOCKET_SHUTDOWN	The network layer was unable to shut down a socket. Check the following: 1) The machine is connected to the network, and the network is responsive. 2) The other side of the connection is running normally. 3) There are enough system resources available. Free up system resources if they are running low.	68
SOCKET_STARTUP	The network layer was unable to initialize the socket layer. Check the following: 1) The sockets layer is properly installed. The correct network interface library must be present and accessible. 2) The machine is connected to the network, and the network is responsive. 3) There are enough system resources available. Free up system resources if they are running low.	70
UNEXPECTED_HTTP_REQUEST_TYPE	The given HTTP request type was unexpected at this time. The most likely cause is an HTTP client that is not a MobiLink client.	234
UNRECOGNIZED_TLS_TYPE	The TLS type is invalid. Consult the documentation for valid types.	240

Member name	Description	Value
VALUE_OUT_OF_RANGE	The specified value was not in the range of acceptable values for that parameter. Check the documentation for the parameter to learn the acceptable range of values.	233
WOULD_BLOCK	A requested operation would block where blocking is undesirable or unexpected.	13
WRITE	Unable to write the given number of bytes to the network layer. Note that writes may occur as part of any larger network operation. For example, some network layers have sub-layers that perform several reads and writes as part of a basic operation in the upper layer. The cause of a write error is usually one of the following: 1) The network had a problem that caused the write to fail. Reconnect and retry the operation. 2) The connection timed out. Reconnect and retry the operation. 3) The other side of the connection cleanly terminated the connection. Consult the client and/or server logs for errors that indicate why the connection has been dropped. Consult the output-log errors and fix the cause, then retry the operation. 4) The process at the other side of the connection was aborted. Consult the client and/or server output logs for errors that indicate why the process was aborted. If the process was shut down by other than normal means, there may not be any errors in its output log. Reconnect and retry the operation. 5) The system is low on resources, and cannot perform the write. Free up system resources, reconnect and retry the operation. If subsequent retry attempts fail, consult your network administrator.	9
WRITE_TIMEOUT	Unable to write the given number of bytes to the network layer in the given time. Check that the network is functioning correctly, and that the receiving application is still running.	202

See also

- ◆ [“StreamErrorCode property” on page 901](#)

ULStreamErrorContext enumeration

UL Ext.: Enumerates the basic network operation being performed when the stream errors occurred.

Syntax

Visual Basic

Public Enum **ULStreamErrorContext**

C#

public enum **ULStreamErrorContext**

Members

Member name	Description	Value
CLOSE	CLOSE(6)	6
CREATE	CREATE(3)	3
DESTROY	DESTROY(4)	4
END_READ	END_READ(11)	11
END_WRITE	END_WRITE(10)	10
GETVALUE	GETVALUE(14)	14
OPEN	OPEN(5)	5
READ	READ(7)	7
REGISTER	REGISTER(1)	1
SOFTSHUTDOWN	SOFTSHUTDOWN(13)	13
UNKNOWN	UNKNOWN(0)	0
UNREGISTER	UNREGISTER(2)	2
WRITE	WRITE(8)	8
WRITE_FLUSH	WRITE_FLUSH(9)	9
YIELD	YIELD(12)	12

See also

- ♦ [“StreamErrorContext property” on page 901](#)

ULStreamErrorID enumeration

UL Ext.: Enumerates the network layers that may report errors during synchronization.

Syntax

Visual Basic

Public Enum **ULStreamErrorID**

C#

public enum **ULStreamErrorID**

Members

Member name	Description	Value
ACTIVESYNC	ACTIVESYNC(22)	22
CERTICOM	CERTICOM(11)	11
CERTICOM_SSL	CERTICOM_SSL(13)	13
CERTICOM_TLS	CERTICOM_TLS(14)	14
DH_CAST	DH_CAST(9)	9
EMAIL	EMAIL(20)	20
FAKE	FAKE(2)	2
FILE	FILE(21)	21
HTTP	HTTP(7)	7
HTTPS	HTTPS(8)	8
JAVA_CERTICOM	JAVA_CERTICOM(12)	12
JAVA_RSA	JAVA_RSA(24)	24
NETTECH	NETTECH(5)	5
OPEN_SSL_RSA	OPEN_SSL_RSA(25)	25
PALM_CONDUIT	PALM_CONDUIT(3)	3
PALM_SS	PALM_SS(4)	4
PALM_SSL	PALM_SSL(26)	26
REPLAY	REPLAY(17)	17
RIMBB	RIMBB(6)	6

Member name	Description	Value
RSA_TLS	RSA_TLS(23)	23
SECURE	SECURE(10)	10
SERIAL	SERIAL(1)	1
STRM	STRM(18)	18
TCPIP	TCPIP(0)	0
UDP	UDP(19)	19
WIRELESS	WIRELESS(16)	16
WIRESTRM	WIRESTRM(15)	15

See also

- ◆ [“StreamErrorID property” on page 902](#)

ULStreamType enumeration

UL Ext.: Enumerates the types of MobiLink synchronization streams to use for synchronization.

Syntax

Visual Basic

Public Enum **ULStreamType**

C#

public enum **ULStreamType**

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams”](#) [*MobiLink - Client Administration*].

Separately licensed component required

ECC encryption and FIPS-certified encryption require a separate license. All strong encryption technologies are subject to export regulations.

See [“Separately licensed components”](#) [*SQL Anywhere 10 - Introduction*].

Members

Member name	Description	Value
HTTP	Synchronize via HTTP. The HTTP stream uses TCP/IP as its underlying transport. UltraLite applications act as Web browsers and the MobiLink server acts as a Web server. UltraLite applications send POST requests to send data to the server and GET requests to read data from the server.	1
HTTPS	Synchronize via HTTPS (HTTP with transport-layer security).	2
TCPIP	Synchronize via TCP/IP.	0
TLS	Synchronize via TCP/IP with transport layer security.	3

See also

- ♦ [“Stream property”](#) on page 878

ULSyncParms class

UL Ext.: Represents synchronization parameters that define how to synchronize an UltraLite database. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULSyncParms**

C#

public sealed class **ULSyncParms**

Remarks

There is no constructor for this class. Each connection has its own ULSyncParms instance, attached as its ULConnection.SyncParms.

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a ULSQLCode.SQLE_SYNC_INFO_INVALID SQLException is thrown by ULConnection.Synchronize().

Other sources of ULSQLCode.SQLE_SYNC_INFO_INVALID errors include not specifying a ULSyncParms.Stream value or a ULSyncParms.Version value.

See also

- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“SyncParms property” on page 553](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“DownloadOnly property” on page 872](#)
- ◆ [“PingOnly property” on page 875](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UploadOnly property” on page 880](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“Stream property” on page 878](#)
- ◆ [“Version property” on page 881](#)

ULSyncParms members

Public properties

Member name	Description
AuthenticationParms property	Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Member name	Description
CheckpointStore property	Specifies whether the client should perform extra store checkpoints to control the growth of the database store during synchronization.
DisableConcurrency property	Specifies whether concurrent access to UltraLite while performing a synchronization.
DownloadOnly property	Specifies whether to disable or enable uploads when synchronizing.
KeepPartialDownload property	Specifies whether to disable or enable partial downloads when synchronizing.
NewPassword property	Specifies a new MobiLink password for the user specified with UserName.
Password property	The MobiLink password for the user specified by UserName.
PingOnly property	Specifies whether the client should only ping the MobiLink server instead of performing a real synchronization.
PublicationMask property	Specifies the publications to be synchronized.
ResumePartialDownload property	Specifies whether to resume or discard a previous partial download.
SendColumnNames property	Specifies whether the client should send column names to the MobiLink server during synchronization.
SendDownloadAck property	Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization. The download acknowledgement is sent after the download has been fully applied and committed at the remote (a positive acknowledgement) or after the download fails (a negative acknowledgement).
Stream property	Specifies the MobiLink synchronization stream to use for synchronization.
StreamParms property	Specifies the parameters to configure the synchronization stream.
TableOrder property	Specifies the order tables should be uploaded to the consolidated database.
UploadOnly property	Specifies whether to disable or enable downloads when synchronizing.
UserName property	The user name that uniquely identifies the MobiLink client to the MobiLink server.
Version property	Specifies which synchronization script to use.

Public methods

Member name	Description
CopyFrom method	Copies the properties of the specified ULSyncParms object to this ULSyncParms object.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [“SyncParms property” on page 553](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“DownloadOnly property” on page 872](#)
- ◆ [“PingOnly property” on page 875](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UploadOnly property” on page 880](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“Stream property” on page 878](#)
- ◆ [“Version property” on page 881](#)

AuthenticationParms property

Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Syntax**Visual Basic**

Public Property **AuthenticationParms** As String

C#

```
public string AuthenticationParms { get; set; }
```

Property value

An array of strings, each containing an authentication parameter (null array entries result in a synchronization error). The default is a null reference (Nothing in Visual Basic), meaning no authentication parameters.

Remarks

Only the first 255 strings are used and each string should be no longer than 128 characters (longer strings are truncated when sent to the MobiLink server).

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

CheckpointStore property

Specifies whether the client should perform extra store checkpoints to control the growth of the database store during synchronization.

Syntax

Visual Basic

Public Property **CheckpointStore** As Boolean

C#

```
public bool CheckpointStore { get; set; }
```

Property value

True to specify that the client should perform extra store checkpoints. The default is false, meaning only required checkpointing is done.

Remarks

The checkpoint operation adds I/O operations for the application, and so slows synchronization. This option is most useful for large downloads with many updates. Devices with slow flash memory might not want to pay the performance penalty associated with additional checkpoints.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

DisableConcurrency property

Specifies whether concurrent access to UltraLite while performing a synchronization.

Syntax

Visual Basic

Public Property **DisableConcurrency** As Boolean

C#

```
public bool DisableConcurrency { get; set; }
```

Property value

True to disable concurrent access to UltraLite while performing a synchronization, false to enable concurrent access. The default is false.

Remarks

By default, other threads can perform UltraLite operations while a thread is synchronizing. When concurrent synchronization is disabled, other threads block on UltraLite calls until synchronization completes.

See also

- ♦ [“ULSyncParms class” on page 868](#)
- ♦ [“ULSyncParms members” on page 868](#)

DownloadOnly property

Specifies whether to disable or enable uploads when synchronizing.

Syntax**Visual Basic**

Public Property **DownloadOnly** As Boolean

C#

```
public bool DownloadOnly { get; set; }
```

Property value

True to disable uploads when synchronizing, false to enable uploads. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

See also

- ♦ [“ULSyncParms class” on page 868](#)
- ♦ [“ULSyncParms members” on page 868](#)
- ♦ [“UploadOnly property” on page 880](#)
- ♦ [“DownloadOnly property” on page 872](#)
- ♦ [“PingOnly property” on page 875](#)
- ♦ [“ResumePartialDownload property” on page 876](#)
- ♦ [“UploadOnly property” on page 880](#)
- ♦ [“ULSQLCode enumeration” on page 836](#)
- ♦ [“Synchronize\(\) method” on page 573](#)

KeepPartialDownload property

Specifies whether to disable or enable partial downloads when synchronizing.

Syntax**Visual Basic**

Public Property **KeepPartialDownload** As Boolean

C#

```
public bool KeepPartialDownload { get; set; }
```

Property value

True to enable partial downloads when synchronizing, false to disable partial downloads. The default is false.

Remarks

UltraLite.NET has the ability to restart downloads that fail because of communication errors or user aborts through the ULSyncProgressListener. UltraLite.NET processes the download as it is received. If a download is interrupted, then the partial download transaction remains in the database and can be resumed during the next synchronization.

To indicate that UltraLite.NET should save partial downloads, specify `connection.SyncParms.KeepPartialDownload=true`; otherwise the download is rolled back if an error occurs.

If a partial download was kept, then the output field `connection.SyncResult.ULSyncResult.PartialDownloadRetained` is set to true when `connection.Synchronize()` exits.

If `PartialDownloadRetained` is set, then you can resume a download. To do this, call `connection.Synchronize()` with `connection.SyncParms.ULSyncParms.ResumePartialDownload` set to true. It is recommended that you keep `KeepPartialDownload` set to true as well in case another communications error occurs. No upload is done if a download is skipped.

The download you receive during a resumed download is as old as when the download originally began. If you need the most up to date data, then you can do another download immediately after the special resumed download completes.

When resuming a download, many of the ULSyncParms fields are not relevant. For example, the `PublicationMask` field is not used. You receive the publications that you requested on the initial download. The only fields that need to be set are `ResumePartialDownload` and `UserName`. The fields `KeepPartialDownload` and `DisableConcurrency` can be set if desired and function as normal.

If you have a partial download and it is no longer needed, then you can call `ULConnection.RollbackPartialDownload()` to roll back the failed download transaction. Also, if you attempt to synchronize again and do not specify `ResumePartialDownload`, then the partial download is rolled back before the next synchronization begins.

For more information, see the [“Resuming failed downloads” \[MobiLink - Server Administration\]](#).

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“PartialDownloadRetained property” on page 900](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“RollbackPartialDownload method” on page 572](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UserName property” on page 881](#)
- ◆ [“DisableConcurrency property” on page 871](#)
- ◆ [“RollbackPartialDownload method” on page 572](#)

NewPassword property

Specifies a new MobiLink password for the user specified with UserName.

Syntax

Visual Basic

Public Property **NewPassword** As String

C#

public string **NewPassword** { get; set; }

Property value

A string specifying a new MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning the password is not changed.

Remarks

A new password takes effect after the next synchronization.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“UserName property” on page 881](#)

Password property

The MobiLink password for the user specified by UserName.

Syntax

Visual Basic

Public Property **Password** As String

C#

public string **Password** { get; set; }

Property value

A string specifying the MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning no password is specified.

Remarks

The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULSyncParms class” on page 868](#)

- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“NewPassword property” on page 874](#)
- ◆ [“UserName property” on page 881](#)

PingOnly property

Specifies whether the client should only ping the MobiLink server instead of performing a real synchronization.

Syntax

Visual Basic

Public Property **PingOnly** As Boolean

C#

```
public bool PingOnly { get; set; }
```

Property value

True to specify that the client should only ping the MobiLink server, false to specify the client should perform a real synchronization. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“DownloadOnly property” on page 872](#)
- ◆ [“PingOnly property” on page 875](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UploadOnly property” on page 880](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“Synchronize\(\) method” on page 573](#)

PublicationMask property

Specifies the publications to be synchronized.

Syntax

Visual Basic

Public Property **PublicationMask** As Integer

C#

```
public int PublicationMask { get; set; }
```

Property value

A bitwise combination of publication masks, the special value `ULPublicationSchema.SYNC_ALL_PUBS`, or the special value `ULPublicationSchema.SYNC_ALL_DB`. The default is `ULPublicationSchema.SYNC_ALL_DB`. For more information on publication masks, see [“ULPublicationSchema class” on page 788](#).

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“Mask property” on page 790](#)
- ◆ [“SYNC_ALL_PUBS field” on page 790](#)
- ◆ [“SYNC_ALL_DB field” on page 789](#)
- ◆ [“SYNC_ALL_DB field” on page 789](#)
- ◆ [“ULPublicationSchema class” on page 788](#)

ResumePartialDownload property

Specifies whether to resume or discard a previous partial download.

Syntax**Visual Basic**

Public Property **ResumePartialDownload** As Boolean

C#

```
public bool ResumePartialDownload { get; set; }
```

Property value

True to resume a previous partial download, false to discard a previous partial download. The default is false.

Remarks

Only at most one synchronization command (`ULSyncParms.DownloadOnly`, `ULSyncParms.PingOnly`, `ULSyncParms.ResumePartialDownload`, or `ULSyncParms.UploadOnly`) can be specified at a time. If more than one of these parameters is set to true, a `ULSQLCode.SQLE_SYNC_INFO_INVALID` `SQLException` is thrown by `ULConnection.Synchronize()`.

For more information on partial downloads, see [“KeepPartialDownload property” on page 872](#).

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“DownloadOnly property” on page 872](#)
- ◆ [“PingOnly property” on page 875](#)

- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UploadOnly property” on page 880](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“Synchronize\(\) method” on page 573](#)
- ◆ [“PartialDownloadRetained property” on page 900](#)

SendColumnNames property

Specifies whether the client should send column names to the MobiLink server during synchronization.

Syntax

Visual Basic

Public Property **SendColumnNames** As Boolean

C#

```
public bool SendColumnNames { get; set; }
```

Property value

True to specify that the client should send column names to the MobiLink server, false to specify that column names are not sent. The default is false.

Remarks

The column names are used by the MobiLink server for direct row handling. When the MobiLink server is using the row handling API to refer to columns by name rather than by index, you should set this option. This is the only use of the column names that are sent by this option.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

SendDownloadAck property

Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization. The download acknowledgement is sent after the download has been fully applied and committed at the remote (a positive acknowledgement) or after the download fails (a negative acknowledgement).

Syntax

Visual Basic

Public Property **SendDownloadAck** As Boolean

C#

```
public bool SendDownloadAck { get; set; }
```

Property value

Set True to specify that the client should send a download acknowledgement to the MobiLink server. Set False to specify that no download acknowledgement is sent. The default is False.

Remarks

If the client sends a download acknowledgement, the MobiLink server database worker thread must wait for the client to apply and commit the download. If the client does not send a download acknowledgement, the MobiLink server is freed up sooner for its next synchronization.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

Stream property

Specifies the MobiLink synchronization stream to use for synchronization.

Syntax**Visual Basic**

Public Property **Stream** As ULStreamType

C#

```
public ULStreamType Stream { get; set; }
```

Property value

One of the ULStreamType values specifying the type of synchronization stream to use. The default is ULStreamType.TCPIP.

Remarks

Most synchronization streams require parameters to identify the MobiLink server address and control other behavior. These parameters are supplied by ULSyncParms.StreamParms.

If the stream type is set to a value that is invalid for the platform, the stream type is set to ULStreamType.TCPIP.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“ULStreamType enumeration” on page 867](#)
- ◆ [“StreamParms property” on page 879](#)
- ◆ [“ULStreamType enumeration” on page 867](#)

StreamParms property

Specifies the parameters to configure the synchronization stream.

Syntax

Visual Basic

Public Property **StreamParms** As String

C#

```
public string StreamParms { get; set; }
```

Property value

A string, in the form of a semicolon-separated list of keyword-value pairs, specifying the parameters for the stream. The default is a null reference (Nothing in Visual Basic).

Remarks

For information on configuring specific stream types, see [“Network protocol options for UltraLite synchronization streams” \[MobiLink - Client Administration\]](#).

StreamParms is a string containing all the parameters used for synchronization streams. Parameters are specified as a semicolon-separated list of name=value pairs ("param1=value1;param2=value2").

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“Stream property” on page 878](#)
- ◆ [“ULStreamType enumeration” on page 867](#)

TableOrder property

Specifies the order tables should be uploaded to the consolidated database.

Syntax

Visual Basic

Public Property **TableOrder** As String

C#

```
public string TableOrder { get; set; }
```

Property value

A string, in the form of a comma-separated list of table names. Tables names can be quoted using either single or double quotes. The default is a null reference (Nothing in Visual Basic), which does not override the default ordering of tables.

Remarks

If the foreign keys on your consolidated database match the foreign keys on your remote UltraLite database and there are no foreign key cycles, then you most likely don't need to use this feature. However, if you have tables that are part of foreign key cycles then list all tables that are part of a cycle in the TableOrder field. If you have tables with different foreign key relationships on the consolidated then also list these tables in the TableOrder field.

All tables that you don't specify will be appropriately sorted based of the foreign keys defined in the remote database.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

UploadOnly property

Specifies whether to disable or enable downloads when synchronizing.

Syntax

Visual Basic

Public Property **UploadOnly** As Boolean

C#

```
public bool UploadOnly { get; set; }
```

Property value

True to disable downloads, false to enable downloads. The default is false.

Remarks

At most, only one synchronization command (ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly) can be specified at a time. If more than one of these parameters is set to true, a ULSQLCode.SQLE_SYNC_INFO_INVALID SQLException is thrown by ULConnection.Synchronize().

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“DownloadOnly property” on page 872](#)
- ◆ [“PingOnly property” on page 875](#)
- ◆ [“ResumePartialDownload property” on page 876](#)
- ◆ [“UploadOnly property” on page 880](#)
- ◆ [“ULSQLCode enumeration” on page 836](#)
- ◆ [“Synchronize\(\) method” on page 573](#)

Username property

The user name that uniquely identifies the MobiLink client to the MobiLink server.

Syntax

Visual Basic

Public Property **UserName** As String

C#

```
public string UserName { get; set; }
```

Property value

A string specifying the user name. This parameter has no default value, and must be explicitly set.

Remarks

The MobiLink server uses this value to determine the download content, to record the synchronization state, and to recover from interruptions during synchronization. This user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“Password property” on page 874](#)

Version property

Specifies which synchronization script to use.

Syntax

Visual Basic

Public Property **Version** As String

C#

```
public string Version { get; set; }
```

Property value

A string specifying the version of the synchronization script to use. This parameter has no default value, and must be explicitly set.

Remarks

Each synchronization script in the consolidated database is marked with a version string. For example, there can be two different download_cursor scripts, with each one identified by a different version string. The version string allows an UltraLite application to choose from a set of synchronization scripts.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)

CopyFrom method

Copies the properties of the specified ULSyncParms object to this ULSyncParms object.

Syntax**Visual Basic**

```
Public Sub CopyFrom( _  
    ByVal src As ULSyncParms _  
)
```

C#

```
public void CopyFrom(  
    ULSyncParms src  
);
```

Parameters

- ◆ **src** The object to copy from.

See also

- ◆ [“ULSyncParms class” on page 868](#)
- ◆ [“ULSyncParms members” on page 868](#)
- ◆ [“ULSyncParms class” on page 868](#)

ULSyncProgressData class

UL Ext.: Returns synchronization progress monitoring data.

Syntax

Visual Basic

Public Class **ULSyncProgressData**

C#

public class **ULSyncProgressData**

See also

- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressListener interface” on page 894](#)

ULSyncProgressData members

Public fields

Member name	Description
FLAG_IS_BLOCKING field	A flag indicating that the synchronization is blocked awaiting a response from the MobiLink server. This field is constant and read-only.

Public properties

Member name	Description
ErrorMessage property	Returns the error message describing the error that occurred during synchronization.
Flags property	Returns the current synchronization flags indicating additional information relating to the current state.
ReceivedBytes property	Returns the number of bytes received so far. This information is updated for all states.
ReceivedDeletes property	Returns the number of deleted rows received so far.
ReceivedInserts property	Returns the number of inserted rows received so far.
ReceivedUpdates property	Returns the number of updated rows received so far.
SQLCode property	Returns the SQL code for synchronization.
SentBytes property	Returns the number of bytes sent so far. This information is updated for all states.

Member name	Description
SentDeletes property	Returns the number of deleted rows sent so far.
SentInserts property	Returns the number of inserted rows sent so far.
SentUpdates property	Returns the number of updated rows sent so far.
State property	Returns the current synchronization state.
SyncParms property	Returns a reference to the connection's SyncParms object.
SyncResult property	Returns a reference to the connection's SyncResult object. This object is only updated with “ ULSyncProgressState enumeration ” on page 896 and “ ULSyncProgressState enumeration ” on page 896 events.
TableCount property	A count of the tables sent or received (TableCount of TableTotal) so far.
TableIndex property	Returns the index of the table currently being synchronized (tables are numbered 1 to DatabaseSchema.TableCount).
TableTotal property	Returns the number of tables being synchronized.

See also

- ◆ “[ULSyncProgressData class](#)” on page 883
- ◆ “[ULSyncProgressListener interface](#)” on page 894

FLAG_IS_BLOCKING field

A flag indicating that the synchronization is blocked awaiting a response from the MobiLink server. This field is constant and read-only.

Syntax**Visual Basic**

```
Public Shared FLAG_IS_BLOCKING As Integer
```

C#

```
public const int FLAG_IS_BLOCKING ;
```

See also

- ◆ “[ULSyncProgressData class](#)” on page 883
- ◆ “[ULSyncProgressData members](#)” on page 883

ErrorMessage property

Returns the error message describing the error that occurred during synchronization.

Syntax

Visual Basic

Public Readonly Property **ErrorMessage** As String

C#

```
public string ErrorMessage { get;}
```

Property value

A string describing the error that occurred during synchronization.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

Flags property

Returns the current synchronization flags indicating additional information relating to the current state.

Syntax

Visual Basic

Public Readonly Property **Flags** As Integer

C#

```
public int Flags { get;}
```

Property value

An integer containing a combination of flags or'ed together.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“FLAG_IS_BLOCKING field” on page 884](#)

ReceivedBytes property

Returns the number of bytes received so far. This information is updated for all states.

Syntax

Visual Basic

Public Readonly Property **ReceivedBytes** As Long

C#

public long **ReceivedBytes** { get;}

Property value

The number of bytes received so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

ReceivedDeletes property

Returns the number of deleted rows received so far.

Syntax

Visual Basic

Public Readonly Property **ReceivedDeletes** As Integer

C#

public int **ReceivedDeletes** { get;}

Property value

The number of deleted rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

ReceivedInserts property

Returns the number of inserted rows received so far.

Syntax**Visual Basic**

Public Readonly Property **ReceivedInserts** As Integer

C#

```
public int ReceivedInserts { get;}
```

Property value

The number of inserted rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

ReceivedUpdates property

Returns the number of updated rows received so far.

Syntax**Visual Basic**

Public Readonly Property **ReceivedUpdates** As Integer

C#

```
public int ReceivedUpdates { get;}
```

Property value

The number of updated rows received so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SQLCode property

Returns the SQL code for synchronization.

Syntax**Visual Basic**

Public Readonly Property **SQLCode** As ULSQLCode

C#

public ULSQLCode **SQLCode** { get;}

Property value

The [“ULSQLCode enumeration” on page 836](#) value for any synchronization error.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SentBytes property

Returns the number of bytes sent so far. This information is updated for all states.

Syntax**Visual Basic**

Public Readonly Property **SentBytes** As Long

C#

public long **SentBytes** { get;}

Property value

The number of bytes sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SentDeletes property

Returns the number of deleted rows sent so far.

Syntax**Visual Basic**

Public Readonly Property **SentDeletes** As Integer

C#

```
public int SentDeletes { get;}
```

Property value

The number of deleted rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SentInserts property

Returns the number of inserted rows sent so far.

Syntax**Visual Basic**

Public Readonly Property **SentInserts** As Integer

C#

```
public int SentInserts { get;}
```

Property value

The number of inserted rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SentUpdates property

Returns the number of updated rows sent so far.

Syntax**Visual Basic**

Public Readonly Property **SentUpdates** As Integer

C#

```
public int SentUpdates { get;}
```

Property value

The number of updated rows sent so far.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

State property

Returns the current synchronization state.

Syntax**Visual Basic**

Public Readonly Property **State** As ULSyncProgressState

C#

```
public ULSyncProgressState State { get;}
```

Property value

One of the [“ULSyncProgressState enumeration” on page 896](#) values specifying the current synchronization state.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

SyncParms property

Returns a reference to the connection's SyncParms object.

Syntax**Visual Basic**

Public Readonly Property **SyncParms** As ULSyncParms

C#

```
public ULSyncParms SyncParms { get;}
```

Property value

A reference to the [“SyncParms property” on page 553](#) object.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“SyncParms property” on page 553](#)

SyncResult property

Returns a reference to the connection's SyncResult object. This object is only updated with [“ULSyncProgressState enumeration” on page 896](#) and [“ULSyncProgressState enumeration” on page 896](#) events.

Syntax**Visual Basic**

Public Readonly Property **SyncResult** As ULSyncResult

C#

public ULSyncResult **SyncResult** { get;}

Property value

A reference to the [“SyncResult property” on page 554](#) object.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

TableCount property

A count of the tables sent or received (TableCount of TableTotal) so far.

Syntax**Visual Basic**

Public Readonly Property **TableCount** As Integer

C#

public int **TableCount** { get;}

Property value

The count of tables sent or received.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

TableIndex property

Returns the index of the table currently being synchronized (tables are numbered 1 to DatabaseSchema.TableCount).

Syntax**Visual Basic**

Public Readonly Property **TableIndex** As Integer

C#

```
public int TableIndex { get;}
```

Property value

The index of the table currently being synchronized. Tables are numbered 1 to [“TableCount property” on page 649](#).

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

TableTotal property

Returns the number of tables being synchronized.

Syntax**Visual Basic**

Public Readonly Property **TableTotal** As Integer

C#

```
public int TableTotal { get;}
```

Property value

The number of tables being synchronized.

See also

- ◆ [“ULSyncProgressData class” on page 883](#)
- ◆ [“ULSyncProgressData members” on page 883](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)
- ◆ [“ULSyncProgressState enumeration” on page 896](#)

ULSyncProgressListener interface

UL Ext.: The listener interface for receiving synchronization progress events.

Syntax

Visual Basic

Public Interface **ULSyncProgressListener**

C#

public interface **ULSyncProgressListener**

See also

- ◆ [“ULSyncProgressListener members” on page 894](#)
- ◆ [“Synchronize\(ULSyncProgressListener\) method” on page 574](#)

ULSyncProgressListener members

Public methods

Member name	Description
SyncProgressed method	Invoked during synchronization to inform the user of progress. This method should return true to cancel synchronization or return false to continue.

See also

- ◆ [“ULSyncProgressListener interface” on page 894](#)
- ◆ [“Synchronize\(ULSyncProgressListener\) method” on page 574](#)

SyncProgressed method

Invoked during synchronization to inform the user of progress. This method should return true to cancel synchronization or return false to continue.

Syntax

Visual Basic

Public Function **SyncProgressed**(
 ByVal *data* As ULSyncProgressData
) As Boolean

C#

public bool **SyncProgressed**(
 ULSyncProgressData *data*
);

Parameters

- ◆ **data** A ULSyncProgressData object containing the latest synchronization progress data.

Return value

This method should return true to cancel synchronization or return false to continue.

Remarks

No UltraLite.NET API methods should be invoked during a SyncProgressed call.

See also

- ◆ [“ULSyncProgressListener interface” on page 894](#)
- ◆ [“ULSyncProgressListener members” on page 894](#)
- ◆ [“ULSyncProgressData class” on page 883](#)

ULSyncProgressState enumeration

UL Ext.: Enumerates all the states that can occur while synchronizing.

Syntax

Visual Basic

Public Enum **ULSyncProgressState**

C#

public enum **ULSyncProgressState**

Members

Member name	Description	Value
STATE_CANCELLED	Synchronization has been cancelled.	15
STATE_COMMITTING_DOWNLOAD	The download is being committed. The final count of rows received is included with this event. See “ReceivedBytes property” on page 885 , “ReceivedInserts property” on page 886 , “ReceivedUpdates property” on page 887 , and “ReceivedDeletes property” on page 886 .	9
STATE_CONNECTING	The synchronization stream has been built, but is not yet opened.	1
STATE_DISCONNECTING	The synchronization stream is about to be closed.	11
STATE_DONE	Synchronization has successfully completed. The connection’s “SyncResult property” on page 891 object has been updated.	12
STATE_ERROR	Synchronization has completed, but an error occurred. Check “SyncResult property” on page 891 , “ErrorMessage property” on page 885 , and “SQL-Code property” on page 887 for details.	13
STATE_FINISHING_UPLOAD	The upload is completing. The final count of rows sent is included with this event. See “SentBytes property” on page 888 , “SentInserts property” on page 889 , “SentUpdates property” on page 889 , and “SentDeletes property” on page 888 .	5
STATE_LAST	The last state entered by synchronization. This state is always entered and always after any other state. Other ULSyncProgressData fields may not contain valid data.	128

Member name	Description	Value
STATE_RECEIVING_DATA	Data for the current table is being received. “ReceivedBytes property” on page 885 , “ReceivedInserts property” on page 886 , “ReceivedUpdates property” on page 887 , and “ReceivedDeletes property” on page 886 have been updated.	8
STATE_RECEIVING_TABLE	A table is being received. Progress can be monitored using “TableIndex property” on page 892 and “TableCount property” on page 891 .	7
STATE_RECEIVING_UPLOAD_ACK	An acknowledgement that the upload is complete is being received.	6
STATE_ROLLING_BACK_DOWNLOAD	Synchronization is rolling back the download because an error was encountered during the download. The error will be reported with a subsequent STATE_ERROR progress report.	14
STATE_SENDING_DATA	Data for the current table is being sent. “SentBytes property” on page 888 , “SentInserts property” on page 889 , “SentUpdates property” on page 889 , and “SentDeletes property” on page 888 have been updated.	4
STATE_SENDING_DOWNLOAD_ACK	An acknowledgement that the download is complete is being sent.	10
STATE_SENDING_HEADER	The synchronization stream has been opened and the header is about to be sent.	2
STATE_SENDING_TABLE	A table is being sent. Progress can be monitored using “TableIndex property” on page 892 and “TableCount property” on page 891 .	3
STATE_STARTING	No synchronization actions have been taken yet.	0

See also

- ◆ [“ULSyncProgressData class” on page 883](#)

ULSyncResult class

UL Ext.: Represents the status of the last synchronization.

Syntax

Visual Basic

Public Class **ULSyncResult**

C#

public class **ULSyncResult**

Remarks

There is no constructor for this class. Each connection has its own ULSyncResult instance, attached as its ULConnection.SyncResult. A ULSyncResult instance is only valid while that connection is open.

See also

- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“SyncResult property” on page 554](#)
- ◆ [“Synchronize\(\) method” on page 573](#)

ULSyncResult members

Public properties

Member name	Description
AuthStatus property	Returns the authorization status code for the last synchronization attempt.
AuthValue property	Returns the return value from custom user authentication synchronization scripts.
IgnoredRows property	Checks whether any uploaded rows were ignored during the last synchronization.
PartialDownloadRetained property	Checks whether a partial download was retained during the last synchronization.
StreamErrorCode property	Returns the error reported by the stream itself.
StreamErrorContext property	Returns the basic network operation being performed when the stream error occurred.
StreamErrorID property	Returns the ID of the network layer reporting an error.
StreamErrorSystem property	Returns the stream error system-specific code.
Timestamp property	Returns the timestamp of the last synchronization.

Member name	Description
UploadOK property	Checks whether the last upload synchronization was successful.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“SyncResult property” on page 554](#)
- ◆ [“Synchronize\(\) method” on page 573](#)

AuthStatus property

Returns the authorization status code for the last synchronization attempt.

Syntax**Visual Basic**

Public Readonly Property **AuthStatus** As ULAuthStatusCode

C#

```
public ULAuthStatusCode AuthStatus { get;}
```

Property value

One of the ULAuthStatusCode values denoting the authorization status for the last synchronization attempt.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“ULAuthStatusCode enumeration” on page 467](#)

AuthValue property

Returns the return value from custom user authentication synchronization scripts.

Syntax**Visual Basic**

Public Readonly Property **AuthValue** As Long

C#

```
public long AuthValue { get;}
```

Property value

A long integer returned from custom user authentication synchronization scripts.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)

IgnoredRows property

Checks whether any uploaded rows were ignored during the last synchronization.

Syntax**Visual Basic**

Public Readonly Property **IgnoredRows** As Boolean

C#

```
public bool IgnoredRows { get;}
```

Property value

True if any uploaded rows were ignored during the last synchronization, false if no rows were ignored.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“DownloadOnly property” on page 872](#)

PartialDownloadRetained property

Checks whether a partial download was retained during the last synchronization.

Syntax**Visual Basic**

Public Readonly Property **PartialDownloadRetained** As Boolean

C#

```
public bool PartialDownloadRetained { get;}
```

Property value

True if a download was interrupted and the partial download was retained, false if the download was not interrupted or if the partial download was rolled back.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“KeepPartialDownload property” on page 872](#)

StreamErrorCode property

Returns the error reported by the stream itself.

Syntax

Visual Basic

Public Readonly Property **StreamErrorCode** As UStreamErrorCode

C#

public UStreamErrorCode **StreamErrorCode** { get;}

Property value

One of the UStreamErrorCode values denoting the error reported by the stream itself, UStreamErrorCode.NONE if no error occurred.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“UStreamErrorCode enumeration” on page 846](#)
- ◆ [“UStreamErrorCode enumeration” on page 846](#)

StreamErrorContext property

Returns the basic network operation being performed when the stream error occurred.

Syntax

Visual Basic

Public Readonly Property **StreamErrorContext** As UStreamErrorContext

C#

public UStreamErrorContext **StreamErrorContext** { get;}

Property value

One of the UStreamErrorContext values denoting which basic network operation was being performed when the stream error occurred.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“UStreamErrorContext enumeration” on page 864](#)

StreamErrorID property

Returns the ID of the network layer reporting an error.

Syntax

Visual Basic

Public Readonly Property **StreamErrorID** As ULStreamErrorID

C#

public ULStreamErrorID **StreamErrorID** { get;}

Property value

One of the ULStreamErrorID values denoting the ID of the network layer reporting an error.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [“ULStreamErrorID enumeration” on page 865](#)

StreamErrorSystem property

Returns the stream error system-specific code.

Syntax

Visual Basic

Public Readonly Property **StreamErrorSystem** As Integer

C#

public int **StreamErrorSystem** { get;}

Property value

An integer denoting the stream error system-specific code.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)

Timestamp property

Returns the timestamp of the last synchronization.

Syntax**Visual Basic**

Public Readonly Property **Timestamp** As Date

C#

public DateTime **Timestamp** { get;}

Property value

A System.DateTime specifying the timestamp of the last synchronization.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)
- ◆ [DateTime](#)

UploadOK property

Checks whether the last upload synchronization was successful.

Syntax**Visual Basic**

Public Readonly Property **UploadOK** As Boolean

C#

public bool **UploadOK** { get;}

Property value

True if the last upload synchronization was successful, false if the last upload synchronization was unsuccessful.

See also

- ◆ [“ULSyncResult class” on page 898](#)
- ◆ [“ULSyncResult members” on page 898](#)

ULTable class

UL Ext.: Represents a table in an UltraLite database.

Syntax

Visual Basic

Public Class **ULTable**
Inherits ULResultSet

C#

public class **ULTable** : ULResultSet

Remarks

There is no constructor for this class. Tables are created using the ULCommand.ExecuteTable() of the ULCommand.

Inherits: ULResultSet

Implements: System.Data.IDataReader, System.Data.IDataRecord, System.IDisposable

See also

- ◆ [“ULTable members” on page 904](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

ULTable members

Public properties

Member name	Description
Depth property (inherited from ULDataReader)	Returns the depth of nesting for the current row. The outermost table has a depth of zero.
FieldCount property (inherited from ULDataReader)	Returns the number of columns in the cursor.
HasRows property (inherited from ULDataReader)	Checks whether the ULDataReader has one or more rows.
IsBOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is before the first row.

Member name	Description
IsClosed property (inherited from ULDataReader)	Checks whether the cursor is currently open.
IsEOF property (inherited from ULDataReader)	UL Ext.: Checks whether the current row position is after the last row.
Item properties (inherited from ULDataReader)	Gets the value of a specified column as an instance of Object .
RecordsAffected property (inherited from ULDataReader)	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement. For SELECT statements or CommandType.TableDirect tables, this value is -1.
RowCount property (inherited from ULDataReader)	UL Ext.: Returns the number of rows in the cursor.
Schema property	Holds the table schema. This property is only valid while its connection is open.
VisibleFieldCount (inherited from DbDataReader)	

Public methods

Member name	Description
AppendBytes method (inherited from ULResultSet)	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.
AppendChars method (inherited from ULResultSet)	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDbType.LongVarchar column.
Close method (inherited from ULDataReader)	Closes the cursor.
Delete method (inherited from ULResultSet)	Deletes the current row.
DeleteAllRows method	Deletes all rows in the table.
Dispose (inherited from DbDataReader)	
FindBegin method	Prepares to perform a new Find on a table.
FindFirst methods	Moves forward through the table from the beginning, looking for a row that exactly matches a value or full set of values in the current index.
FindLast methods	Moves backward through the table from the end, looking for a row that exactly matches a value or full set of values in the current index.

Member name	Description
FindNext methods	Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or full set of values in the current index.
FindPrevious methods	Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or full set of values in the current index.
GetBoolean method (inherited from ULDataReader)	Returns the value for the specified column as a System.Boolean.
GetByte method (inherited from ULDataReader)	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
GetBytes methods (inherited from ULDataReader)	UL Ext.: Returns the value for the specified column as an array of System.Bytes. Only valid for columns of type ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier.
GetChar method (inherited from ULDataReader)	This method is not supported in UltraLite.NET.
GetChars method (inherited from ULDataReader)	Copies a subset of the value for the specified ULDbType.LongVarChar column, beginning at the specified offset, to the specified offset of the destination System.Char array.
GetData (inherited from DbDataReader)	
GetDataTypeName method (inherited from ULDataReader)	Returns the name of the specified column's provider data type.
GetDateTime method (inherited from ULDataReader)	Returns the value for the specified column as a System.DateTime with millisecond accuracy.
GetDecimal method (inherited from ULDataReader)	Returns the value for the specified column as a System.Decimal.
GetDouble method (inherited from ULDataReader)	Returns the value for the specified column as a System.Double.
GetEnumerator method (inherited from ULDataReader)	Returns an System.Collections.IEnumerator that iterates through the ULDataReader.
GetFieldType method (inherited from ULDataReader)	Returns the System.Type most appropriate for the specified column.
GetFloat method (inherited from ULDataReader)	Returns the value for the specified column as a System.Single.
GetGuid method (inherited from ULDataReader)	Returns the value for the specified column as a UUID (System.Guid).

Member name	Description
GetInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.Int16.
GetInt32 method (inherited from ULDataReader)	Returns the value for the specified column as an Int32.
GetInt64 method (inherited from ULDataReader)	Returns the value for the specified column as an Int64.
GetName method (inherited from ULDataReader)	Returns the name of the specified column.
GetOrdinal method (inherited from ULDataReader)	Returns the column ID of the named column.
GetProviderSpecificFieldType (inherited from DbDataReader)	
GetProviderSpecificValue (inherited from DbDataReader)	
GetProviderSpecificValues (inherited from DbDataReader)	
GetSchemaTable method (inherited from ULDataReader)	Returns a System.Data.DataTable that describes the column metadata of the ULDataReader.
GetString method (inherited from ULDataReader)	Returns the value for the specified column as a System.String.
GetTimeSpan method (inherited from ULDataReader)	Returns the value for the specified column as a System.TimeSpan with millisecond accuracy.
GetUInt16 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt16.
GetUInt32 method (inherited from ULDataReader)	Returns the value for the specified column as a UInt32.
GetUInt64 method (inherited from ULDataReader)	Returns the value for the specified column as a System.UInt64.
GetValue method (inherited from ULDataReader)	Returns the value of the specified column in its native format.
GetValues method (inherited from ULDataReader)	Returns all the column values for the current row.
Insert method	Inserts a new row with the current column values (specified using the set methods). Each insert must be preceded by a call to ULTable.InsertBegin.

Member name	Description
InsertBegin method	Prepares to insert a new row into the table by setting all current column values to their default values.
IsDBNull method (inherited from ULDataReader)	Checks whether the value from the specified column is NULL.
LookupBackward methods	Moves backward through the table from the end, looking for a row that matches or is less than a value or full set of values in the current index.
LookupBegin method	Prepares to perform a new lookup on the table. The value(s) for which to search are specified by calling the appropriate setType method(s) on the columns in the index with which the table was opened.
LookupForward methods	Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or full set of values in the current index.
MoveAfterLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to after the last row of the cursor.
MoveBeforeFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to before the first row of the cursor.
MoveFirst method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the first row of the cursor.
MoveLast method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the last row of the cursor.
MoveNext method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
MovePrevious method (inherited from ULDataReader)	UL Ext.: Positions the cursor to the previous row or before the first row.
MoveRelative method (inherited from ULDataReader)	UL Ext.: Positions the cursor relative to the current row.
NextResult method (inherited from ULDataReader)	Advances the ULDataReader to the next result when reading the results of batch SQL statements.
Read method (inherited from ULDataReader)	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
SetBoolean method (inherited from ULResultSet)	Sets the value for the specified column using a System.Boolean .
SetByte method (inherited from ULResultSet)	Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).
SetBytes method (inherited from ULResultSet)	Sets the value for the specified column using an array of System.Bytes .

Member name	Description
SetDBNull method (inherited from ULResultSet)	Sets a column to NULL.
SetDateTime method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.DateTime</code> .
SetDecimal method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.Decimal</code> .
SetDouble method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.Double</code> .
SetFloat method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.Single</code> .
SetGuid method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.Guid</code> .
SetInt16 method (inherited from ULResultSet)	Sets the value for the specified column using an <code>System.Int16</code> .
SetInt32 method (inherited from ULResultSet)	Sets the value for the specified column using an <code>System.Int32</code> .
SetInt64 method (inherited from ULResultSet)	Sets the value for the specified column using an <code>Int64</code> .
SetString method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.String</code> .
SetTimeSpan method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.TimeSpan</code> .
SetToDefault method (inherited from ULResultSet)	Sets the value for the specified column to its default value.
SetUInt16 method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.UInt16</code> .
SetUInt32 method (inherited from ULResultSet)	Sets the value for the specified column using an <code>System.UInt32</code> .
SetUInt64 method (inherited from ULResultSet)	Sets the value for the specified column using a <code>System.UInt64</code> .
Truncate method	Deletes all rows in the table while temporarily activating a stop synchronization delete.
Update method (inherited from ULResultSet)	Updates the current row with the current column values (specified using the set methods).
UpdateBegin method	Prepares to update the current row in the table.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ExecuteTable\(\) method” on page 529](#)
- ◆ [“ULCommand class” on page 497](#)
- ◆ [“ULResultSet class” on page 792](#)
- ◆ [IDataReader](#)
- ◆ [IDataRecord](#)
- ◆ [IDisposable](#)

Schema property

Holds the table schema. This property is only valid while its connection is open.

Syntax**Visual Basic**

Public Readonly Property **Schema** As ULTableSchema

C#

public ULTableSchema **Schema** { get;}

Property value

The ULTableSchema object representing the table schema.

Remarks

This property represents the complete schema of the table, including UltraLite.NET extended information which is not represented in the results from `ULDataReader.GetSchemaTable`.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“ULTableSchema class” on page 925](#)

DeleteAllRows method

Deletes all rows in the table.

Syntax**Visual Basic**

Public Sub **DeleteAllRows()**

C#

public void **DeleteAllRows();**

Remarks

In some applications, it can be useful to delete all rows from a table before downloading a new set of data into the table. Rows can be deleted from the UltraLite database without being deleted from the consolidated database using `ULConnection.StopSynchronizationDelete`.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“Truncate method” on page 923](#)
- ◆ [“StopSynchronizationDelete method” on page 573](#)

FindBegin method

Prepares to perform a new Find on a table.

Syntax**Visual Basic**

```
Public Sub FindBegin()
```

C#

```
public void FindBegin();
```

Remarks

The value(s) for which to search are specified by calling the appropriate `setType` method(s) on the columns in the index with which the table was opened.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindFirst\(\) method” on page 911](#)
- ◆ [“FindFirst\(Int16\) method” on page 912](#)
- ◆ [“FindLast\(\) method” on page 913](#)
- ◆ [“FindLast\(Int16\) method” on page 914](#)

FindFirst methods

Moves forward through the table from the beginning, looking for a row that exactly matches a value or full set of values in the current index.

FindFirst() method

Moves forward through the table from the beginning, looking for a row that exactly matches a value or full set of values in the current index.

Syntax**Visual Basic**

Public Function **FindFirst()** As Boolean

C#

public bool **FindFirst()**;

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to FindBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindFirst methods” on page 911](#)
- ◆ [“FindBegin method” on page 911](#)
- ◆ [“FindNext\(\) method” on page 915](#)
- ◆ [“FindPrevious\(\) method” on page 917](#)
- ◆ [“FindFirst\(Int16\) method” on page 912](#)
- ◆ [“IsEOF property” on page 662](#)
- ◆ [“FindBegin method” on page 911](#)

FindFirst(Int16) method

Moves forward through the table from the beginning, looking for a row that exactly matches a value or partial set of values in the current index.

Syntax**Visual Basic**

Public Function **FindFirst**(
 ByVal *numColumns* As Short
) As Boolean

C#

public bool **FindFirst**(
 short *numColumns*
);

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to FindBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindFirst methods” on page 911](#)
- ◆ [“FindBegin method” on page 911](#)
- ◆ [“FindNext\(Int16\) method” on page 916](#)
- ◆ [“FindPrevious\(Int16\) method” on page 917](#)
- ◆ [“FindFirst\(\) method” on page 911](#)
- ◆ [“IsEOF property” on page 662](#)
- ◆ [“FindBegin method” on page 911](#)

FindLast methods

Moves backward through the table from the end, looking for a row that exactly matches a value or full set of values in the current index.

FindLast() method

Moves backward through the table from the end, looking for a row that exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Public Function **FindLast()** As Boolean

C#

public bool **FindLast();**

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row found that exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to FindBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindLast methods” on page 913](#)
- ◆ [“FindBegin method” on page 911](#)
- ◆ [“FindNext\(\) method” on page 915](#)
- ◆ [“FindPrevious\(\) method” on page 917](#)
- ◆ [“FindLast\(Int16\) method” on page 914](#)
- ◆ [“IsBOF property” on page 661](#)
- ◆ [“FindBegin method” on page 911](#)

FindLast(Int16) method

Moves backward through the table from the end, looking for a row that exactly matches a value or partial set of values in the current index.

Syntax**Visual Basic**

```
Public Function FindLast( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindLast(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to find a value that matches based on the first column only, you should set the value for the first column, then supply a value of 1.

Return value

True if successful, false otherwise

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row found that exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to FindBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindLast methods” on page 913](#)
- ◆ [“FindBegin method” on page 911](#)
- ◆ [“FindNext\(Int16\) method” on page 916](#)
- ◆ [“FindPrevious\(Int16\) method” on page 917](#)
- ◆ [“FindLast\(\) method” on page 913](#)
- ◆ [“IsBOF property” on page 661](#)
- ◆ [“FindBegin method” on page 911](#)

FindNext methods

Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or full set of values in the current index.

FindNext() method

Continues a ULTable.FindFirst() search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

Public Function **FindNext()** As Boolean

C#

public bool **FindNext();**

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the next row if it exactly matches the index value. On failure, the cursor position is after the last row (ULDataReader.IsEOF).

FindNext behavior is undefined if the column values being searched for are modified during a row update.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindNext methods” on page 915](#)
- ◆ [“FindFirst\(\) method” on page 911](#)
- ◆ [“FindNext\(Int16\) method” on page 916](#)
- ◆ [“IsEOF property” on page 662](#)

FindNext(Int16) method

Continues a `ULTable.FindFirst()` search by moving forward through the table from the current position, looking to see if the next row exactly matches a value or partial set of values in the current index.

Syntax**Visual Basic**

```
Public Function FindNext( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindNext(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index, and you want to find a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the next row if it exactly matches the index value. On failure, the cursor position is after the last row (`ULDataReader.IsEOF`).

`FindNext` behavior is undefined if the column values being searched for are modified during a row update.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindNext methods” on page 915](#)
- ◆ [“FindFirst\(Int16\) method” on page 912](#)
- ◆ [“FindNext\(\) method” on page 915](#)
- ◆ [“FindFirst\(\) method” on page 911](#)
- ◆ [“IsEOF property” on page 662](#)

FindPrevious methods

Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or full set of values in the current index.

FindPrevious() method

Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or full set of values in the current index.

Syntax

Visual Basic

```
Public Function FindPrevious() As Boolean
```

C#

```
public bool FindPrevious();
```

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the previous row if it exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

FindPrevious behavior is undefined if the column values being searched for are modified during a row update.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindPrevious methods” on page 917](#)
- ◆ [“FindLast\(\) method” on page 913](#)
- ◆ [“FindPrevious\(Int16\) method” on page 917](#)
- ◆ [“IsBOF property” on page 661](#)

FindPrevious(Int16) method

Continues a ULTable.FindLast() search by moving backward through the table from the current position, looking to see if the previous row exactly matches a value or partial set of values in the current index.

Syntax

Visual Basic

```
Public Function FindPrevious( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool FindPrevious(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the find. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

The cursor is left on the previous row if it exactly matches the index value. On failure, the cursor position is before the first row (ULDataReader.IsBOF).

FindPrevious behavior is undefined if the column values being searched for are modified during a row update.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“FindPrevious methods” on page 917](#)
- ◆ [“FindLast\(\) method” on page 913](#)
- ◆ [“FindLast\(Int16\) method” on page 914](#)
- ◆ [“FindPrevious\(\) method” on page 917](#)
- ◆ [“IsBOF property” on page 661](#)

Insert method

Inserts a new row with the current column values (specified using the set methods).

Each insert must be preceded by a call to ULTable.InsertBegin.

Syntax**Visual Basic**

```
Public Sub Insert()
```

C#

```
public void Insert();
```

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“InsertBegin method” on page 919](#)

InsertBegin method

Prepares to insert a new row into the table by setting all current column values to their default values.

Syntax

Visual Basic

```
Public Sub InsertBegin()
```

C#

```
public void InsertBegin();
```

Remarks

Call the appropriate SetType or AppendType method(s) to specify the non-default values that are to be inserted.

The row is not actually inserted and the data in the row is not actually changed until you execute the Insert method, and that change is not made permanent until it is committed.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“Insert method” on page 918](#)
- ◆ [“Insert method” on page 918](#)

LookupBackward methods

Moves backward through the table from the end, looking for a row that matches or is less than a value or full set of values in the current index.

LookupBackward() method

Moves backward through the table from the end, looking for a row that matches or is less than a value or full set of values in the current index.

Syntax

Visual Basic

```
Public Function LookupBackward() As Boolean
```

C#

```
public bool LookupBackward();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is less than the index value. On failure, (no rows less than the value being looked for) the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to LookupBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“LookupBackward methods” on page 919](#)
- ◆ [“LookupBegin method” on page 921](#)
- ◆ [“LookupBackward\(Int16\) method” on page 920](#)
- ◆ [“IsBOF property” on page 661](#)

LookupBackward(Int16) method

Moves backward through the table from the beginning, looking for a row that matches or is less than a value or partial set of values in the current index.

Syntax

Visual Basic

```
Public Function LookupBackward( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool LookupBackward(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the lookup. For example, if you have a three column index, and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is less than the index value. On failure, (no rows less than the value being looked for) the cursor position is before the first row (ULDataReader.IsBOF).

Each search must be preceded by a call to LookupBegin.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“LookupBackward methods” on page 919](#)
- ◆ [“LookupBegin method” on page 921](#)
- ◆ [“LookupBackward\(Int16\) method” on page 920](#)
- ◆ [“IsBOF property” on page 661](#)

LookupBegin method

Prepares to perform a new lookup on the table. The value(s) for which to search are specified by calling the appropriate setType method(s) on the columns in the index with which the table was opened.

Syntax**Visual Basic**

```
Public Sub LookupBegin()
```

C#

```
public void LookupBegin();
```

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“LookupForward\(\) method” on page 921](#)
- ◆ [“LookupForward\(Int16\) method” on page 922](#)
- ◆ [“LookupBackward\(\) method” on page 919](#)
- ◆ [“LookupBackward\(Int16\) method” on page 920](#)

LookupForward methods

Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or full set of values in the current index.

LookupForward() method

Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or full set of values in the current index.

Syntax**Visual Basic**

```
Public Function LookupForward() As Boolean
```

C#

```
public bool LookupForward();
```

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is greater than the index value. On failure, (no rows greater than the value being looked for) the cursor position is after the last row (ULDataReader.IsEOF).

Each search must be preceded by a call to ULTable.LookupBegin().

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“LookupForward methods” on page 921](#)
- ◆ [“LookupBegin method” on page 921](#)
- ◆ [“LookupForward\(Int16\) method” on page 922](#)
- ◆ [“IsEOF property” on page 662](#)

LookupForward(Int16) method

Moves forward through the table from the beginning, looking for a row that matches or is greater than a value or partial set of values in the current index.

Syntax**Visual Basic**

```
Public Function LookupForward( _  
    ByVal numColumns As Short _  
) As Boolean
```

C#

```
public bool LookupForward(  
    short numColumns  
);
```

Parameters

- ◆ **numColumns** For composite indexes, the number of columns to use in the lookup. For example, if you have a three column index and you want to look up a value that matches based on the first column only, you should set the value for the first column, and then supply a value of 1.

Return value

True if successful, false otherwise.

Remarks

To specify the value for which to search, set the column value for each column in the index. The cursor is left on the first row that matches or is greater than the index value. On failure, (no rows greater than the value being looked for) the cursor position is after the last row `ULDataReader.IsEOF`).

Each search must be preceded by a call to `LookupBegin`.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“LookupForward methods” on page 921](#)
- ◆ [“LookupBegin method” on page 921](#)
- ◆ [“LookupForward\(\) method” on page 921](#)
- ◆ [“IsEOF property” on page 662](#)
- ◆ [“LookupBegin method” on page 921](#)

Truncate method

Deletes all rows in the table while temporarily activating a stop synchronization delete.

Syntax

Visual Basic

```
Public Sub Truncate()
```

C#

```
public void Truncate();
```

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“DeleteAllRows method” on page 910](#)

UpdateBegin method

Prepares to update the current row in the table.

Syntax

Visual Basic

```
Public Sub UpdateBegin()
```

C#

```
public void UpdateBegin();
```

Remarks

Column values are modified by calling the appropriate setType or AppendType method(s). The first append on a column clears the current column value prior to appending the new value.

The data in the row is not actually changed until you call ULResultSet.Update(), and that change is not made permanent until it is committed.

Modifying columns in the index used to open the table affects any active searches in unpredictable ways. Columns in the primary key of the table can not be updated.

See also

- ◆ [“ULTable class” on page 904](#)
- ◆ [“ULTable members” on page 904](#)
- ◆ [“Update method” on page 815](#)

ULTableSchema class

UL Ext.: Represents the schema of an UltraLite table. This class cannot be inherited.

Syntax

Visual Basic

```
Public NotInheritable Class ULTableSchema
    Inherits ULCursorSchema
```

C#

```
public sealed class ULTableSchema : ULCursorSchema
```

Remarks

There is no constructor for this class. A ULTableSchema object is attached to a table as its ULTable.Schema.

Inherits: ULCursorSchema

See also

- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“Schema property” on page 910](#)
- ◆ [“ULCursorSchema class” on page 618](#)

ULTableSchema members

Public properties

Member name	Description
ColumnCount property (inherited from ULCursorSchema)	Returns the number of columns in the cursor.
IndexCount property	Returns the number of indexes on the table.
IsNeverSynchronized property	Checks whether the table is marked as never being synchronized.
IsOpen property (inherited from ULCursorSchema)	Checks whether the cursor schema is currently open.
Name property	Returns the name of the table.
PrimaryKey property	Returns the index schema of the primary key for the table.
UploadUnchangedRows property	Checks whether the database uploads rows that have not changed.

Public methods

Member name	Description
GetColumnDefaultValue method	Returns the default value of the specified column.
GetColumnID method (inherited from ULCursorSchema)	Returns the column ID of the named column.
GetColumnName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnPartitionSize method	Returns the global autoincrement partition size assigned to the specified column.
GetColumnPrecision method (inherited from ULCursorSchema)	Returns the precision of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSQLName method (inherited from ULCursorSchema)	Returns the name of the column identified by the specified column ID.
GetColumnScale method (inherited from ULCursorSchema)	Returns the scale of the column identified by the specified column ID if the column is a numeric column (SQL type NUMERIC).
GetColumnSize method (inherited from ULCursorSchema)	Returns the size of the column identified by the specified column ID if the column is a sized column (SQL type BINARY or CHAR).
GetColumnULDbType method (inherited from ULCursorSchema)	Returns the UltraLite.NET data type of the column identified by the specified column ID.
GetIndex method	Returns the index schema of the named index.
GetIndexName method	Returns the name of the index identified by the specified index ID.
GetOptimalIndex method	The optimal index for searching a table using the specified column.
GetPublicationPredicate method	Returns the publication predicate for this table in the named publication.
GetSchemaTable method (inherited from ULCursorSchema)	Returns a System.Data.DataTable that describes the column schema of the ULDataReader.
IsColumnAutoIncrement method	Checks whether the specified column's default is set to autoincrement.
IsColumnCurrentDate method	Checks whether the specified column's default is set to the current date (ULDbType.Date).
IsColumnCurrentTime method	Checks whether the specified column's default is set to the current time (ULDbType.Time).

Member name	Description
IsColumnCurrentTimestamp method	Checks whether the specified column's default is set to the current timestamp (ULDbType.TimeStamp).
IsColumnGlobalAutoIncrement method	Checks whether the specified column's default is set to global autoincrement.
IsColumnNewUUID method	Checks whether the specified column's default is set to a new UUID (System.Guid).
IsColumnNullable method	Checks whether the specified column is nullable.
IsInPublication method	Checks whether the table is contained in the named publication.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“Schema property” on page 910](#)
- ◆ [“ULCursorSchema class” on page 618](#)

IndexCount property

Returns the number of indexes on the table.

Syntax**Visual Basic**

Public Readonly Property **IndexCount** As Integer

C#

```
public int IndexCount { get;}
```

Property value

The number of indexes on the table or 0 if the table schema is closed.

Remarks

Index IDs range from 1 to IndexCount, inclusively.

Note: Index IDs and count may change during a schema upgrade. To correctly identify an index, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

IsNeverSynchronized property

Checks whether the table is marked as never being synchronized.

Syntax

Visual Basic

Public Readonly Property **IsNeverSynchronized** As Boolean

C#

```
public bool IsNeverSynchronized { get;}
```

Property value

True if the table is marked as never being synchronized, false otherwise.

Remarks

Tables marked as never being synchronized are never synchronized, even if they are included in a publication. These tables are sometimes referred to as "no sync" tables.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

Name property

Returns the name of the table.

Syntax

Visual Basic

Public Overrides Readonly Property **Name** As String

C#

```
public override string Name { get;}
```

Property value

The name of the table as a string.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

PrimaryKey property

Returns the index schema of the primary key for the table.

Syntax**Visual Basic**

Public Readonly Property **PrimaryKey** As ULIndexSchema

C#

public ULIndexSchema **PrimaryKey** { get;}

Property value

A ULIndexSchema object representing the primary key for the table.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ULIndexSchema class” on page 729](#)

UploadUnchangedRows property

Checks whether the database uploads rows that have not changed.

Syntax**Visual Basic**

Public Readonly Property **UploadUnchangedRows** As Boolean

C#

public bool **UploadUnchangedRows** { get;}

Property value

True if the table is marked to always upload all rows during synchronization, false if the table is marked to upload only changed rows.

Remarks

Tables marked as such upload unchanged rows, as well as changed rows, when the table is synchronized. These tables are sometimes referred to as "all sync" tables.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

GetColumnDefaultValue method

Returns the default value of the specified column.

Syntax

Visual Basic

```
Public Function GetColumnDefaultValue( _  
    ByVal columnID As Integer _  
) As String
```

C#

```
public string GetColumnDefaultValue(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in a table has an ID value of zero.

Return value

The default value of the specified column as a string or a null reference (Nothing in Visual Basic) if the default value is null.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)

GetColumnPartitionSize method

Returns the global autoincrement partition size assigned to the specified column.

Syntax

Visual Basic

```
Public Function GetColumnPartitionSize( _  
    ByVal columnID As Integer _  
) As UInt64
```

C#

```
public ulong GetColumnPartitionSize(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

The column's global autoincrement partition size as a System.UInt64.

Remarks

All global autoincrement columns in a given table share the same global autoincrement partition.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“IsColumnGlobalAutoIncrement method” on page 936](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [UInt64](#)

GetIndex method

Returns the index schema of the named index.

Syntax**Visual Basic**

```
Public Function GetIndex( _  
    ByVal name As String _  
) As ULIndexSchema
```

C#

```
public ULIndexSchema GetIndex(  
    string name  
);
```

Parameters

- ◆ **name** The name of the index.

Return value

A ULIndexSchema object representing the named index.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ULIndexSchema class” on page 729](#)

GetIndexName method

Returns the name of the index identified by the specified index ID.

Syntax**Visual Basic**

```
Public Function GetIndexName( _
```

```
ByVal indexID As Integer _  
) As String
```

C#

```
public string GetIndexName(  
    int indexID  
);
```

Parameters

- ◆ **indexID** The ID of the index. The value must be in the range [1,IndexCount].

Return value

The name of the index as a string.

Remarks

Index IDs and counts may change during a schema upgrade. To correctly identify an index, access it by name or refresh the cached IDs and counts after a schema upgrade.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“IndexCount property” on page 927](#)

GetOptimalIndex method

The optimal index for searching a table using the specified column.

Syntax

Visual Basic

```
Public Function GetOptimalIndex( _  
    ByVal columnID As Integer _  
) As ULIndexSchema
```

C#

```
public ULIndexSchema GetOptimalIndex(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount>-1]. The first column in the table has an ID value of zero.

Return value

A ULIndexSchema object representing the optimal index for the specified column.

Remarks

The specified column is the first column in the index, but the index may have more than one column.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ULIndexSchema class” on page 729](#)

GetPublicationPredicate method

Returns the publication predicate for this table in the named publication.

Syntax**Visual Basic**

```
Public Function GetPublicationPredicate( _  
    ByVal pubName As String _  
) As String
```

C#

```
public string GetPublicationPredicate(  
    string pubName  
);
```

Parameters

- ◆ **pubName** The name of the publication.

Return value

The publication predicate as a string.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

IsColumnAutoIncrement method

Checks whether the specified column's default is set to autoincrement.

Syntax**Visual Basic**

```
Public Function IsColumnAutoIncrement( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnAutoIncrement(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is autoincrementing, false if it is not autoincrementing.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)

IsColumnCurrentDate method

Checks whether the specified column's default is set to the current date (ULDbType.Date).

Syntax**Visual Basic**

```
Public Function IsColumnCurrentDate( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentDate(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current date, false if the column does not default to the current date.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ULDbType enumeration” on page 695](#)

IsColumnCurrentTime method

Checks whether the specified column's default is set to the current time (ULDbType.Time).

Syntax

Visual Basic

```
Public Function IsColumnCurrentTime( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentTime(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current time, false if the column does not default to the current time.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [“ULDbType enumeration” on page 695](#)

IsColumnCurrentTimestamp method

Checks whether the specified column's default is set to the current timestamp (ULDbType.TimeStamp).

Syntax

Visual Basic

```
Public Function IsColumnCurrentTimestamp( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnCurrentTimestamp(  
    int columnID  
);
```

Parameters

- ♦ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to the current timestamp, false if the column does not default to the current timestamp.

See also

- ♦ [“ULTableSchema class” on page 925](#)
- ♦ [“ULTableSchema members” on page 925](#)
- ♦ [“ColumnCount property” on page 619](#)
- ♦ [“ULDbType enumeration” on page 695](#)

IsColumnGlobalAutoIncrement method

Checks whether the specified column's default is set to global autoincrement.

Syntax**Visual Basic**

```
Public Function IsColumnGlobalAutoIncrement( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnGlobalAutoIncrement(  
    int columnID  
);
```

Parameters

- ♦ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is global autoincrementing, false if it is not global autoincrementing.

See also

- ♦ [“ULTableSchema class” on page 925](#)
- ♦ [“ULTableSchema members” on page 925](#)
- ♦ [“GetColumnPartitionSize method” on page 930](#)
- ♦ [“DatabaseID property” on page 550](#)
- ♦ [“ColumnCount property” on page 619](#)

IsColumnNewUUID method

Checks whether the specified column's default is set to a new UUID (System.Guid).

Syntax

Visual Basic

```
Public Function IsColumnNewUUID( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnNewUUID(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column defaults to a new UUID, false if the column does not default to a new UUID.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)
- ◆ [Guid](#)

IsColumnNullable method

Checks whether the specified column is nullable.

Syntax

Visual Basic

```
Public Function IsColumnNullable( _  
    ByVal columnID As Integer _  
) As Boolean
```

C#

```
public bool IsColumnNullable(  
    int columnID  
);
```

Parameters

- ◆ **columnID** The ID number of the column. The value must be in the range [0,ULCursorSchema.ColumnCount-1]. The first column in the table has an ID value of zero.

Return value

True if the column is nullable, false if it is not nullable.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)
- ◆ [“ColumnCount property” on page 619](#)

IsInPublication method

Checks whether the table is contained in the named publication.

Syntax**Visual Basic**

```
Public Function IsInPublication( _  
    ByVal pubName As String _  
) As Boolean
```

C#

```
public bool IsInPublication(  
    string pubName  
);
```

Parameters

- ◆ **pubName** The name of the publication.

Return value

True if the table is in the publication, false if the table is not in the publication.

See also

- ◆ [“ULTableSchema class” on page 925](#)
- ◆ [“ULTableSchema members” on page 925](#)

ULTransaction class

Represents a SQL transaction. This class cannot be inherited.

Syntax

Visual Basic

Public NotInheritable Class **ULTransaction**
Inherits DbTransaction

C#

public sealed class **ULTransaction** : DbTransaction

Remarks

There is no constructor for ULTransaction. To obtain a ULTransaction object, use the ULConnection.BeginTransaction(). To associate a command with a transaction, use the ULCommand.Transaction.

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

Inherits: System.Data.Common.DbTransaction

Implements: System.Data.IDbTransaction, System.IDisposable

See also

- ◆ [“ULTransaction members” on page 939](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [DbTransaction](#)
- ◆ [IDbTransaction](#)
- ◆ [IDisposable](#)

ULTransaction members

Public properties

Member name	Description
Connection property	Returns the connection associated with the transaction.
IsolationLevel property	Returns the isolation level for the transaction.

Public methods

Member name	Description
Commit method	Commits the database transaction.

Member name	Description
Dispose (inherited from Db-Transaction)	
Rollback method	Rolls back the transaction's outstanding changes to the database.

See also

- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [“Transaction property” on page 509](#)
- ◆ [DbTransaction](#)
- ◆ [IDbTransaction](#)
- ◆ [IDisposable](#)

Connection property

Returns the connection associated with the transaction.

Syntax**Visual Basic**

Public Readonly Property **Connection** As ULConnection

C#

```
public ULConnection Connection { get; }
```

Property value

The ULConnection object associated with the transaction, or a null reference (Nothing in Visual Basic) if the transaction is no longer valid.

Remarks

This is the strongly-typed version of System.Data.IDbTransaction.Connection and System.Data.Common.DbCommand.Connection.

See also

- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“ULTransaction members” on page 939](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [“ULConnection class” on page 542](#)
- ◆ [IDbTransaction.Connection](#)
- ◆ [DbCommand.Connection](#)

IsolationLevel property

Returns the isolation level for the transaction.

Syntax

Visual Basic

Public Overrides Readonly Property **IsolationLevel** As IsolationLevel

C#

public override IsolationLevel **IsolationLevel** { get;}

Property value

One of the System.Data.IsolationLevel values. UltraLite.NET only supports System.Data.IsolationLevel.ReadUncommitted.

See also

- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“ULTransaction members” on page 939](#)
- ◆ [“BeginTransaction\(\) method” on page 555](#)
- ◆ [IsolationLevel](#)
- ◆ [IsolationLevel.ReadUncommitted](#)

Commit method

Commits the database transaction.

Syntax

Visual Basic

Public Overrides Sub **Commit()**

C#

public override void **Commit();**

Remarks

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

If Commit() fails due to a database error (for example, a referential integrity error), the transaction remains active. Correct the error and call the Commit() method again or call the ULTransaction.Rollback() to complete the transaction.

See also

- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“ULTransaction members” on page 939](#)
- ◆ [“Rollback method” on page 942](#)

Rollback method

Rolls back the transaction's outstanding changes to the database.

Syntax

Visual Basic

Public Overrides Sub **Rollback()**

C#

public override void **Rollback();**

Remarks

Once a transaction has been committed or rolled back, the connection reverts to automatically committing all operations as they are executed. To group more operations together, a new transaction must be created.

See also

- ◆ [“ULTransaction class” on page 939](#)
- ◆ [“ULTransaction members” on page 939](#)
- ◆ [“Commit method” on page 941](#)

Index

A

- Abort property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 820
- AcceptChangesDuringFill property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 158
- accessing data
 - UltraLite.NET Table API, 22
- accessing schema information
 - UltraLite.NET about, 29
- ActiveSync synchronization
 - UltraLite.NET, 33
- ActiveSyncInvoked method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 53
- ActiveSyncInvoked method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 463
- Add method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 297, 298, 299, 300, 301, 302
- Add methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 489, 773
- Add(Int32, Int32) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 489
- Add(Int32, String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 490
- Add(Object) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 773
- Add(String, Int32) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 491
- Add(String, Object) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 775
- Add(String, String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 492
- Add(String, ULDbType) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 776
- Add(String, ULDbType, Int32) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 777
- Add(String, ULDbType, Int32, String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 778
- Add(ULBulkCopyColumnMapping) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 489
- Add(ULParameter) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 774

- adding database objects using the SQL Anywhere Explorer for UltraLite
 - about, 11
- AdditionalParms property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 124
- AdditionalParms property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 581
- AddRange methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 779
- AddRange(Array) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 779
- AddRange(ULParameter[]) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 780
- AppendBytes method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 321
- AppendBytes method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 797
- AppendChars method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 322
- AppendChars method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 798
- architecture
 - UltraLite.Net, 5
- Authenticating users
 - UltraLite.NET development, 31
- AuthenticationParms property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 248, 387
- AuthenticationParms property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 712, 870
- AuthStatus property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 249, 414
- AuthStatus property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 711, 899
- AuthValue property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 249, 414
- AuthValue property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 712, 899
- AutoCommit mode
 - UltraLite.NET development, 28

B

- BatchSize property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 472
- BeginExecuteNonQuery methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 510
- BeginExecuteNonQuery() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 510

BeginExecuteNonQuery(AsyncCallback, Object) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 510
BeginExecuteReader methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 511
BeginExecuteReader() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 511
BeginExecuteReader(AsyncCallback, Object) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 513
BeginExecuteReader(AsyncCallback, Object, CommandBehavior) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 514
BeginExecuteReader(CommandBehavior) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 512
BeginTransaction method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 102, 103
BeginTransaction methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 554
BeginTransaction() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 555
BeginTransaction(IsolationLevel) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 555
benefits
 UltraLite.NET, 2
bugs
 providing feedback, xvii
BulkCopyTimeout property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 473
BytesReceived property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 261
BytesReceived property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 725

C

CacheSize property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 127
CacheSize property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 583, 595
Cancel method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 69
Cancel method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 515
CanCreateDataSourceEnumerator property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 705
CaseSensitive property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 136
CaseSensitive property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 610
casting
 UltraLite.NET data types in, 24
ChangeDatabase method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 104
ChangeDatabase method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 556
ChangeEncryptionKey method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 105
ChangeEncryptionKey method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 557
ChangePassword method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 558
CheckpointStore property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 387
CheckpointStore property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 871
ChecksumLevel property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 136
ChecksumLevel property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 610
Clear method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 304
Clear method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 781
Close method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 105, 204
Close method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 475, 558, 666
CollationName property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 186
CollationName property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 647
ColumnCount property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 145, 266
ColumnCount property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 619, 730
ColumnMappings property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 473
columns
 UltraLite.NET modification of values , 24
Columns property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 743

Command property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 346, 351

Command property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 825, 830

CommandText property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 63

CommandText property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 503

CommandTimeout property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 64

CommandTimeout property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 504

CommandType property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 64

CommandType property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 505

commit method
 UltraLite.NET transactions, 28

Commit method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 458

Commit method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 941

committing
 UltraLite.NET transactions, 28

configuring the SQL Anywhere Explorer
 UltraLite about, 10

connecting
 UltraLite.NET tutorial, 41

Connection class
 UltraLite.NET, 15

Connection property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 65, 457

Connection property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 506, 940

ConnectionString property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 128

ConnectionString property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 584, 595

connections
 SQL Anywhere Explorer for UltraLite, 9
 UltraLite.NET databases, 15

ConnectionString property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 95

ConnectionString property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 547

ConnectionTimeout property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 96

ConnectionTimeout property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 548

Contains method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 304, 305

Contains method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 492

Contains methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 781

Contains(Object) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 781

Contains(String) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 782

ContainsKey method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 603

ContinueUpdateOnError property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 158

conventions
 documentation, xii
 file names in documentation, xiv

CopyFrom method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 398

CopyFrom method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 882

copying
 UltraLite database objects in Visual Studio .NET, 11

CopyTo method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 305

CopyTo method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 493, 782

Count property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 295

Count property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 769

CountUploadRows method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 106, 107

CountUploadRows methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 559

CountUploadRows(Int32, Int64) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 560

CountUploadRows(Int32, UInt32) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 559

CreateCommand method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 108

CreateCommand method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 560, 706

CreateCommandBuilder method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 706
- CreateConnection method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 706
- CreateConnectionStringBuilder method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 707
- CreateDataAdapter method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 707
- CreateDatabase method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 179
- CreateDatabase method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 641
- CreateParameter method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 70
- CreateParameter method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 515, 708

D

- data manipulation

- UltraLite.NET Table API, 22

- UltraLite.NET with SQL , 18

- data types

- UltraLite.NET API accessing in , 23

- UltraLite.NET conversion of, 24

- DataAdapter property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 84

- DataAdapter property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 536

- Database property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 97

- Database property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 549

- database schemas

- UltraLite.NET access, 29

- DatabaseID property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 97

- DatabaseID property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 550

- DatabaseKey property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 596

- DatabaseManager class

- UltraLite.NET, 15

- DatabaseManager property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 98

- DatabaseManager property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 550

- DatabaseName property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 596

- DatabaseOnCE property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 129

- DatabaseOnCE property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 585, 597

- DatabaseOnDesktop property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 129

- DatabaseOnDesktop property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 585, 598

- DataSource property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 549

- DataSourceInformation property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 744

- DataTypes property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 744

- DateFormat property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 137

- DateFormat property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 611

- DateOrder property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 137

- DateOrder property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 611

- DbType property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 285

- DbType property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 759

- Delete method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 324

- Delete method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 800

- DeleteAllRows method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 425

- DeleteAllRows method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 910

- DeleteCommand property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 159

- DeleteCommand property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 632

- deleting

- UltraLite UltraLite.NET table rows, 26

- deploying

- UltraLite.NET, 48

- Depth property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 198

- Depth property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 660

- DesignTimeVisible property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 66

- DesignTimeVisible property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 506
- DestinationColumn property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 484
- DestinationFileName property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 249
- DestinationFileName property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 713
- DestinationOrdinal property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 485
- DestinationPath property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 250
- DestinationPath property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 713
- DestinationTableName property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 474
- developer community
 - newsgroups, xvii
- development
 - UltraLite.NET, 13
- development platforms
 - UltraLite.NET, 4
- Direction property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 285
- Direction property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 759
- DisableConcurrency property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 388
- DisableConcurrency property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 871
- Dispose method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 205, 459
- Dispose method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 476
- DML
 - UltraLite.NET, 18
- documentation
 - conventions, xii
 - SQL Anywhere, x
- DownloadedFile property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 251
- DownloadedFile property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 714
- DownloadFile method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 257, 258
- DownloadFile methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 721
- DownloadFile() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 721

- DownloadFile(ULFileTransferProgressListener) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 722
- DownloadOnly property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 388
- DownloadOnly property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 872
- DropDatabase method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 181
- DropDatabase method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 642
- dynamic SQL
 - UltraLite.NET tutorial, 43

E

- encryption
 - UltraLite.NET development, 17
- EncryptionKey property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 130
- EncryptionKey property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 586
- EndExecuteNonQuery method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 516
- EndExecuteReader method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 519
- EquivalentTo method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 603
- error handling
 - UltraLite.NET, 30
- ErrorMessage property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 400
- ErrorMessage property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 885
- errors
 - UltraLite.NET handling , 30
- ExecuteNonQuery method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 71
- ExecuteNonQuery method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 522
- ExecuteReader method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 72, 73
- ExecuteReader methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 523
- ExecuteReader() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 523

- ExecuteReader(CommandBehavior) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 524
 - ExecuteResultSet method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 74, 75
 - ExecuteResultSet methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 526
 - ExecuteResultSet() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 526
 - ExecuteResultSet(CommandBehavior) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 526
 - ExecuteScalar method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 77
 - ExecuteScalar method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 528
 - ExecuteTable method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 78, 79, 108, 109, 111
 - ExecuteTable methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 529, 561
 - ExecuteTable() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 529
 - ExecuteTable(CommandBehavior) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 530
 - ExecuteTable(String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 561
 - ExecuteTable(String, String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 562
 - ExecuteTable(String, String, CommandBehavior) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 564
 - Explorer for UltraLite (see SQL Anywhere Explorer for UltraLite)
- F**
- feedback
 - documentation, xvii
 - providing, xvii
 - FieldCount property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 199
 - FieldCount property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 660
 - FileAuthCode property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 251
 - FileAuthCode property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 714
 - FileName property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 251
 - FileName property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 715
 - FileSize property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 261
 - FileSize property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 725
 - FileTransferProgressed method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 263
 - FileTransferProgressed method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 727
 - Fill method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 164, 165, 166, 167
 - FillError event [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 176
 - FillSchema method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 168, 170, 171
 - Finalize method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 146, 205, 324, 418, 426
 - find methods
 - UltraLite.NET, 24
 - find mode
 - UltraLite.NET, 23
 - FindBegin method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 426
 - FindBegin method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 911
 - FindFirst method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 427, 428
 - FindFirst methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 911
 - FindFirst() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 911
 - FindFirst(Int16) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 912
 - finding out more and providing feed back
 - technical support, xvii
 - FindLast method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 429, 430
 - FindLast methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 913
 - FindLast() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 913

FindLast(Int16) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 914
FindNext method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 431
FindNext methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 915
FindNext() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 915
FindNext(Int16) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 916
FindPrevious method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 432, 433
FindPrevious methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 917
FindPrevious() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 917
FindPrevious(Int16) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 917
FIPS property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 138
FIPS property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 611
FLAG_IS_BLOCKING field [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 260, 400
FLAG_IS_BLOCKING field [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 724, 884
Flags property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 261, 401
Flags property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 726, 885
ForceDownload property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 252
ForceDownload property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 715
ForeignKeys property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 745

G

GetBoolean method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 205
GetBoolean method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 666
GetByte method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 206
GetByte method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 667
GetBytes method [UL .NET 1.0]

 iAnywhere.Data.UltraLite namespace, 207, 209
GetBytes methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 667
GetBytes(Int32) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 669
GetBytes(Int32, Int64, Byte[], Int32, Int32) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 668
GetChar method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 209
GetChar method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 670
GetChars method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 210
GetChars method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 670
GetColumnDefaultValue method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 445
GetColumnDefaultValue method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 929
GetColumnID method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 146
GetColumnID method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 620
GetColumnName method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 147, 271
GetColumnName method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 621, 735
GetColumnPartitionSize method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 446
GetColumnPartitionSize method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 930
GetColumnPrecision method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 148
GetColumnPrecision method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 622
GetColumnScale method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 149
GetColumnScale method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 624
GetColumnSize method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 150
GetColumnSize method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 624
GetColumnSQLName method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 150
GetColumnSQLName method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 623

- GetColumnULDbType method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 151
- GetColumnULDbType method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 625
- GetData method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 212
- GetDatabaseProperty method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 188
- GetDatabaseProperty method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 650
- GetDataTypeName method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 213
- GetDataTypeName method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 672
- GetDateTime method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 213
- GetDateTime method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 672
- GetDecimal method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 214
- GetDecimal method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 673
- GetDeleteCommand method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 85
- GetDeleteCommand methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 536
- GetDeleteCommand() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 537
- GetDeleteCommand(Boolean) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 536
- GetDouble method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 215
- GetDouble method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 674
- GetEnumerator method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 306
- GetEnumerator method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 674, 783
- GetFieldType method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 216
- GetFieldType method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 675
- GetFillParameters method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 172
- GetFillParameters method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 636
- GetFloat method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 217
- GetFloat method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 676
- GetGuid method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 218
- GetGuid method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 676
- GetIndex method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 447
- GetIndex method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 931
- GetIndexName method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 447
- GetIndexName method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 931
- GetInsertCommand method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 86
- GetInsertCommand methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 538
- GetInsertCommand() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 539
- GetInsertCommand(Boolean) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 538
- GetInt16 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 219
- GetInt16 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 677
- GetInt32 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 219
- GetInt32 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 678
- GetInt64 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 220
- GetInt64 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 678
- GetLastDownloadTime method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 113
- GetLastDownloadTime method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 565
- GetName method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 221
- GetName method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 679
- GetNewUUID method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 113
- GetNewUUID method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 566
- GetObjectData method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 245

GetData method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 702
 GetOptimalIndex method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 448
 GetOptimalIndex method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 932
 GetOrdinal method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 222
 GetOrdinal method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 680
 GetPublicationName method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 191
 GetPublicationName method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 652
 GetPublicationPredicate method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 449
 GetPublicationPredicate method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 933
 GetPublicationSchema method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 191
 GetPublicationSchema method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 652
 GetSchema methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 567
 GetSchema() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 567
 GetSchema(String) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 567
 GetSchema(String, String[]) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 568
 GetSchemaTable method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 152, 223
 GetSchemaTable method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 625, 681
 GetShortName method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 604
 GetString method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 225
 GetString method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 683
 GetTableCountInPublications method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 192
 GetTableCountInPublications method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 653
 GetTableName method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 193
 GetTableName method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 654
 GetTimeSpan method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 226
 GetTimeSpan method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 683
 getting help
 technical support, xvii
 GetUInt16 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 227
 GetUInt16 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 684
 GetUInt32 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 227
 GetUInt32 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 685
 GetUInt64 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 228
 GetUInt64 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 685
 GetUpdateCommand method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 87
 GetUpdateCommand methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 539
 GetUpdateCommand() method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 540
 GetUpdateCommand(Boolean) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 539
 GetValue method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 229
 GetValue method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 686
 GetValues method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 230
 GetValues method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 687
 GlobalAutoIncrementUsage property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 99
 GlobalAutoIncrementUsage property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 551
 grantConnectTo method
 UltraLite.NET development, 31
 GrantConnectTo method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 114
 GrantConnectTo method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 570

H

HasRows property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 660

help

technical support, xvii

I

iAnywhere developer community

newsgroups, xvii

iAnywhere.Data.UltraLite namespace

about, 2

iAnywhere.Data.UltraLite namespace (.NET 1.0)

iAnywhere.Data.UltraLite namespace, 2

iAnywhere.Data.UltraLite namespace (.NET 2.0)

iAnywhere.Data.UltraLite namespace, 2

iAnywhere.Data.UltraLite namespace [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 52

iAnywhere.Data.UltraLite namespace [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 462

icons

used in manuals, xv

IgnoredRows property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 415

IgnoredRows property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 900

IndexColumns property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 746

IndexCount property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 443

IndexCount property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 927

indexes

UltraLite.NET schema information in , 29

Indexes property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 746

IndexName property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 66

IndexName property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 507

IndexOf method [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 306, 307

IndexOf method [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 493

IndexOf methods [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 783

IndexOf(Object) method [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 783

IndexOf(String) method [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 784

IndexSchema class

UltraLite.NET development, 29

InfoMessage event [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 119

InfoMessage event [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 575

Insert method [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 308, 434

Insert method [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 785, 918

insert mode

UltraLite.NET, 23

InsertBegin method [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 434

InsertBegin method [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 919

InsertCommand property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 160

InsertCommand property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 633

inserting

UltraLite.NET table rows, 26

install-dir

documentation usage, xiv

installing

SQL Anywhere Explorer for UltraLite, 9

Instance field [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 705

Interactive SQL

opening from Visual Studio .NET, 9

INVALID_DATABASE_ID field [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 95

INVALID_DATABASE_ID field [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 547

IsBOF property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 199

IsBOF property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 661

IsCaseSensitive property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 186

IsCaseSensitive property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 647

IsClosed property [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 200

IsClosed property [UL .NET 2.0]

iAnywhere.Data.UltraLite namespace, 661

IsColumnAutoIncrement method [UL .NET 1.0]

iAnywhere.Data.UltraLite namespace, 449

IsColumnAutoIncrement method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 933	IsForeignKeyCheckOnCommit property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 731
IsColumnCurrentDate method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 450	IsForeignKeyNullable property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 267
IsColumnCurrentDate method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 934	IsForeignKeyNullable property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 732
IsColumnCurrentTime method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 451	IsInPublication method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 454
IsColumnCurrentTime method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 935	IsInPublication method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 938
IsColumnCurrentTimestamp method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 452	IsNeverSynchronized property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 443
IsColumnCurrentTimestamp method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 935	IsNeverSynchronized property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 928
IsColumnDescending method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 272	IsNullable property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 286
IsColumnDescending method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 736	IsNullable property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 760
IsColumnGlobalAutoIncrement method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 452	IsolationLevel property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 457
IsColumnGlobalAutoIncrement method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 936	IsolationLevel property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 941
IsColumnNewUUID method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 453	IsOpen property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 145, 187, 268, 314
IsColumnNewUUID method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 937	IsOpen property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 620, 648, 732, 790
IsColumnNullable method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 454	IsPrimaryKey property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 268
IsColumnNullable method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 937	IsPrimaryKey property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 732
IsDBNull method [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 231	IsReadOnly property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 770
IsDBNull method [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 688	IsSynchronized property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 771
IsEOF property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 200	IsUniqueIndex property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 269
IsEOF property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 662	IsUniqueIndex property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 733
IsFixedSize property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 770	IsUniqueKey property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 269
IsForeignKey property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 266	IsUniqueKey property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 733
IsForeignKey property [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 731	Item properties [UL .NET 2.0] iAnywhere.Data.UltraLite namespace, 662, 771
IsForeignKeyCheckOnCommit property [UL .NET 1.0] iAnywhere.Data.UltraLite namespace, 267	Item property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 201, 296
- Item property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 488, 598
- Item(Int32) property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 662, 771
- Item(String) property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 663, 772

K

- KeepPartialDownload property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 389
- KeepPartialDownload property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 872

L

- LastIdentity property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 99
- LastIdentity property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 551
- locking
 - UltraLite.NET keywords for, 33
- lookup methods
 - UltraLite.NET, 24
- lookup mode
 - UltraLite.NET, 23
- LookupBackward method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 435, 436
- LookupBackward methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 919
- LookupBackward() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 919
- LookupBackward(Int16) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 920
- LookupBegin method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 437
- LookupBegin method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 921
- LookupForward method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 437, 438
- LookupForward methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 921
- LookupForward() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 921
- LookupForward(Int16) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 922

M

- managing
 - UltraLite.NET transactions, 28
- Mask property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 314
- Mask property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 790
- MaxHashSize property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 138
- MaxHashSize property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 612
- Message property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 273
- Message property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 738
- MetaDataCollections property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 747
- MissingMappingAction property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 160
- MissingSchemaAction property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 161
- modes
 - UltraLite.NET, 23
- MoveAfterLast method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 232
- MoveAfterLast method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 688
- MoveBeforeFirst method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 232
- MoveBeforeFirst method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 689
- moveFirst method
 - UltraLite.NET data retrieval example, 20
- moveFirst method (Table class)
 - UltraLite.NET development, 22
- MoveFirst method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 232
- MoveFirst method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 689
- MoveLast method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 233
- MoveLast method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 690
- moveNext method
 - UltraLite.NET data retrieval example, 20
- moveNext method (Table class)
 - UltraLite.NET development, 22

MoveNext method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 233

MoveNext method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 690

MovePrevious method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 234

MovePrevious method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 691

MoveRelative method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 234

MoveRelative method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 691

multi-threaded applications
 UltraLite.NET overview, 15

N

Name property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 146, 270, 314, 343, 444

Name property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 620, 734, 791, 817, 928

NativeError property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 244, 274

NativeError property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 701, 739

navigating
 UltraLite.NET Table API, 22

navigating SQL result sets
 UltraLite.NET, 20

NearestCentury property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 139

NearestCentury property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 613

NewPassword property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 390

NewPassword property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 874

newsgroups
 technical support, xvii

NextResult method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 235

NextResult method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 692

NotifyAfter property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 475

O

Obfuscate property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 139

Obfuscate property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 613

obfuscation
 UltraLite.NET development, 17

Offset property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 287

Offset property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 760

offsets
 UltraLite.NET relative, 22

online books
 PDF, x

Open method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 115

Open method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 570

options dialog
 SQL Anywhere Explorer for UltraLite, 10

OrderedTableScans property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 599

P

PageSize property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 140

PageSize property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 614

ParameterName property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 287

ParameterName property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 761

Parameters property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 67

Parameters property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 507

PartialDownloadRetained property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 415

PartialDownloadRetained property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 900

Password property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 130, 253, 391

Password property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 586, 600, 716, 874

passwords

- UltraLite.NET authentication in , 31

PDF

- documentation, x

PingOnly property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 391

PingOnly property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 875

Plan property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 68

Plan property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 508

platforms

- supported in UltraLite.NET, 4

Precision property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 140, 288

Precision property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 614, 762

Prepare method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 80

Prepare method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 531

PrimaryKey property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 444

PrimaryKey property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 928

PublicationCount property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 187

PublicationCount property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 648

PublicationMask property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 392

PublicationMask property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 875

publications

- UltraLite.NET schema information in , 29

Publications property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 748

PublicationSchema class

- UltraLite.NET development, 29

Q

QuotePrefix property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 85

QuoteSuffix property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 85

R

Read method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 235

Read method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 692

ReceivedBytes property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 401

ReceivedBytes property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 885

ReceivedDeletes property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 402

ReceivedDeletes property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 886

ReceivedInserts property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 402

ReceivedInserts property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 886

ReceivedUpdates property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 402

ReceivedUpdates property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 887

RecordsAffected property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 202, 346

RecordsAffected property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 664, 826

ReferencedIndexName property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 270

ReferencedIndexName property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 734

ReferencedTableName property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 271

ReferencedTableName property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 735

RefreshSchema method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 88

relative offset

- UltraLite.NET Table API, 22

Remove method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 309

Remove method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 494, 604, 785

RemoveAt method [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 309, 310

RemoveAt method [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 495

RemoveAt methods [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 786
- RemoveAt(Int32) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 786
- RemoveAt(String) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 787
- ReservedWords property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 748
- ReserveSize property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 601
- ResetDbType method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 766
- ResetLastDownloadTime method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 115
- ResetLastDownloadTime method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 571
- Restrictions property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 749
- result set schemas
 - UltraLite.NET, 21
- result sets
 - UltraLite.NET navigation of, 20
- ResumeAtSize property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 262
- ResumeAtSize property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 726
- ResumePartialDownload property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 253, 392
- ResumePartialDownload property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 717, 876
- RevokeConnectFrom method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 116
- RevokeConnectFrom method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 571
- revokeConnectionFrom method
 - UltraLite.NET development, 31
- rollback method
 - UltraLite.NET transactions, 28
- Rollback method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 459
- Rollback method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 942
- RollbackPartialDownload method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 116
- RollbackPartialDownload method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 572
- rollbacks
 - UltraLite.NET transactions, 28
- RowCount property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 203
- RowCount property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 664
- rows
 - UltraLite deletions with UltraLite.NET, 26
 - UltraLite insertions with UltraLite.NET, 26
 - UltraLite updates with UltraLite.NET, 25
 - UltraLite.NET table navigation, 22
 - UltraLite.NETtable access of current, 23
- RowsCopied property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 820
- RowUpdated event [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 176
- RowUpdated event [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 637
- RowUpdating event [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 177
- RowUpdating event [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 638
- RuntimeType property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 179
- RuntimeType property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 640

S

- samples-dir
 - documentation usage, xiv
- Scale property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 141, 289
- Scale property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 614, 762
- Schema property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 100, 204, 425
- Schema property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 552, 665, 910
- schemas
 - UltraLite.NET access, 29
- scrolling
 - UltraLite.NETTable API, 22
- searching
 - UltraLite rows with UltraLite.NET, 24
- SELECT statement
 - UltraLite.NET example, 20
- SelectCommand property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 162

- SelectCommand property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 634
- selecting
 - UltraLite.NET rows , 20
- selecting data from database tables
 - UltraLite.NET, 20
- SendColumnNames property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 393
- SendColumnNames property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 877
- SendDownloadAck property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 394
- SendDownloadAck property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 877
- SentBytes property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 403
- SentBytes property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 888
- SentDeletes property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 403
- SentDeletes property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 888
- SentInserts property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 404
- SentInserts property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 889
- SentUpdates property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 404
- SentUpdates property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 889
- ServerSyncInvoked method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 354
- ServerSyncInvoked method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 833
- ServerVersion property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 552
- SetActiveSyncListener method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 182
- SetActiveSyncListener method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 643
- SetBoolean method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 324
- SetBoolean method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 800
- SetByte method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 325
- SetByte method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 801
- SetBytes method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 326
- SetBytes method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 802
- SetDatabaseOption method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 193
- SetDatabaseOption method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 654
- SetDateTime method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 327
- SetDateTime method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 803
- SetDBNull method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 328
- SetDBNull method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 803
- SetDecimal method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 329
- SetDecimal method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 804
- SetDouble method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 330
- SetDouble method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 805
- SetFloat method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 331
- SetFloat method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 806
- SetGuid method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 332
- SetGuid method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 807
- SetInt16 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 333
- SetInt16 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 808
- SetInt32 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 333
- SetInt32 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 808
- SetInt64 method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 334
- SetInt64 method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 809
- SetServerSyncListener method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 183
- SetServerSyncListener method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 644

SetString method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 335
 SetString method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 810
 SetTimeSpan method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 336
 SetTimeSpan method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 811
 SetToDefault method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 337
 SetToDefault method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 812
 SetUInt16 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 338
 SetUInt16 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 812
 SetUInt32 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 339
 SetUInt32 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 813
 SetUInt64 method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 340
 SetUInt64 method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 814
 SignalSyncIsComplete method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 184
 SignalSyncIsComplete method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 645
 Size property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 289
 Size property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 763
 Source property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 244, 274
 Source property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 701, 739
 SourceColumn property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 290
 SourceColumn property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 485, 763
 SourceColumnNullMapping property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 764
 SourceOrdinal property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 486
 SourceVersion property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 291
 SourceVersion property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 764
 SQL Anywhere
 documentation, x
 SQL Anywhere Explorer
 UltraLite limitations with, 14
 UltraLite supported programming languages, 9
 UltraLite.NET about, 14
 SQL Anywhere Explorer for UltraLite
 about, 7
 adding database objects, 11
 configuring, 10
 connection, 9
 Visual Studio integration, 9
 working with tables, 11
 SQLCode property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 405
 SQLCode property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 887
 StartLine property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 601
 StartSynchronizationDelete method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 117
 StartSynchronizationDelete method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 572
 State property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 100, 405
 State property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 553, 890
 StateChange event [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 120
 StateChange event [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 576
 StopSynchronizationDelete method [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 117
 StopSynchronizationDelete method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 573
 Stream property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 254, 394
 Stream property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 717, 878
 StreamErrorCode property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 255, 415
 StreamErrorCode property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 718, 901
 StreamErrorContext property [UL .NET 1.0]
 iAnywhere.Data.UltraLite namespace, 416
 StreamErrorContext property [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 901
 StreamErrorID property [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 416
- StreamErrorID property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 902
- StreamErrorSystem property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 255, 417
- StreamErrorSystem property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 718, 902
- StreamParms property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 255, 395
- StreamParms property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 719, 879
- support
 - newsgroups, xvii
- supported platforms
 - UltraLite.NET, 4
- Sybase Central
 - opening from Visual Studio .NET, 9
- SYNC_ALL_DB field [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 313
- SYNC_ALL_DB field [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 789
- SYNC_ALL_PUBS field [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 313
- SYNC_ALL_PUBS field [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 790
- synchronization
 - ActiveSync in UltraLite.NET, 33
 - UltraLite.NET, 32
 - UltraLite.NET C# example, 32
- Synchronize method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 118, 119
- Synchronize methods [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 573
- Synchronize() method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 573
- Synchronize(ULSyncProgressListener) method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 574
- SyncParms property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 101, 406
- SyncParms property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 553, 890
- SyncProgressed method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 409
- SyncProgressed method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 894
- SyncResult property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 102, 406

- SyncResult property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 554, 891
- SyncRoot property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 773

T

- Table API
 - UltraLite.NET introduction, 22
- TableCount property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 188, 407
- TableCount property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 649, 891
- TableIndex property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 407
- TableIndex property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 892
- TableMappings property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 162
- TableMappings property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 635
- TableOrder property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 395
- TableOrder property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 879
- tables
 - UltraLite.NET schema information in , 29
- Tables property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 749
- TableSchema class
 - UltraLite.NET development, 29
- TableTotal property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 408
- TableTotal property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 892
- target platforms
 - UltraLite.NET, 4
- technical support
 - newsgroups, xvii
- threads
 - UltraLite.NET multi-threaded applications, 15
- TimeFormat property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 141
- TimeFormat property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 615
- Timestamp property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 417
- Timestamp property [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 902
- TimestampFormat property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 142
- TimestampFormat property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 615
- TimestampIncrement property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 142
- TimestampIncrement property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 616
- ToString method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 132, 143, 274, 292
- ToString method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 588, 617, 740, 766
- transaction processing
 - UltraLite.NET management of, 28
- Transaction property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 68
- Transaction property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 509
- transactions
 - UltraLite.NET management of, 28
- troubleshooting
 - newsgroups, xvii
 - UltraLite.NET deployment checklist, 48
 - UltraLite.NET handling errors, 30
- Truncate method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 439
- Truncate method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 923
- TryGetValue method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 605
- tutorials
 - C# in UltraLite.NET, 35
 - Visual Basic in UltraLite.NET, 35

U

- ULActiveSyncListener interface [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 53
- ULActiveSyncListener interface [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 463
- ULActiveSyncListener members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 463
- ULAuthStatusCode enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 56
- ULAuthStatusCode enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 467
- ULBulkCopy class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 468
- ULBulkCopy constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 469
- ULBulkCopy members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 468
- ULBulkCopy(String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 470
- ULBulkCopy(String, ULBulkCopyOptions) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 471
- ULBulkCopy(ULConnection) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 469
- ULBulkCopy(ULConnection, ULBulkCopyOptions, ULTransaction) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 471
- ULBulkCopyColumnMapping class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 480
- ULBulkCopyColumnMapping constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 481
- ULBulkCopyColumnMapping members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 480
- ULBulkCopyColumnMapping() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 481
- ULBulkCopyColumnMapping(Int32, Int32) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 481
- ULBulkCopyColumnMapping(Int32, String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 482
- ULBulkCopyColumnMapping(String, Int32) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 483
- ULBulkCopyColumnMapping(String, String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 483
- ULBulkCopyColumnMappingCollection class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 487
- ULBulkCopyColumnMappingCollection members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 487
- ULBulkCopyOptions enumeration [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 496
- ULCommand class
 - UltraLite.NET data manipulation example, 18
 - UltraLite.NET data retrieval example, 20
- ULCommand class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 57
- ULCommand class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 497
- ULCommand constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 60, 61, 62
- ULCommand constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 500
- ULCommand members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 498
- ULCommand() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 500
- ULCommand(String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 501
- ULCommand(String, ULConnection) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 502
- ULCommand(String, ULConnection, ULTransaction) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 503
- ULCommandBuilder class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 82
- ULCommandBuilder class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 532
- ULCommandBuilder constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 83, 84
- ULCommandBuilder constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 535
- ULCommandBuilder members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 533
- ULCommandBuilder() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 535
- ULCommandBuilder(ULDataAdapter) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 535
- ULConnection class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 90
- ULConnection class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 542
- ULConnection constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 93, 94
- ULConnection constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 545
- ULConnection members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 542
- ULConnection() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 545
- ULConnection(String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 546
- ULConnectionParms class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 122
- ULConnectionParms class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 578
- ULConnectionParms constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 124
- ULConnectionParms constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 581
- ULConnectionParms members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 579
- ULConnectionParms.UnusedEventHandler delegate [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 133
- ULConnectionParms.UnusedEventHandler delegate [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 589
- ULConnectionStringBuilder class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 590
- ULConnectionStringBuilder constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 594
- ULConnectionStringBuilder members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 591
- ULConnectionStringBuilder() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 594
- ULConnectionStringBuilder(String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 594
- ULCreateParms class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 134
- ULCreateParms class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 607
- ULCreateParms constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 135
- ULCreateParms constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 609
- ULCreateParms members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 608
- ULCursorSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 144
- ULCursorSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 618

- ULCursorSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 618
- ULDataAdapter class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 153
- ULDataAdapter class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 627
- ULDataAdapter constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 155, 156, 157
- ULDataAdapter constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 630
- ULDataAdapter members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 627
- ULDataAdapter() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 630
- ULDataAdapter(String, String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 632
- ULDataAdapter(String, ULConnection) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 631
- ULDataAdapter(ULCommand) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 630
- ULDatabaseManager class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 178
- ULDatabaseManager class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 639
- ULDatabaseManager members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 639
- ULDatabaseSchema class
 - iAnywhere.Data.UltraLite namespace, 29
- ULDatabaseSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 185
- ULDatabaseSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 646
- ULDatabaseSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 646
- ULDataReader class
 - UltraLite.NET data retrieval example, 20
- ULDataReader class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 195
- ULDataReader class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 656
- ULDataReader members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 657
- ULDateOrder enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 237
- ULDateOrder enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 694
- ULDbType enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 238
- ULDbType enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 695
- ULDbType property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 291
- ULDbType property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 765
- ULException class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 242
- ULException class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 699
- ULException constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 243
- ULException constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 700
- ULException members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 699
- ULFactory class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 703
- ULFactory members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 704
- ULFileTransfer class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 246
- ULFileTransfer class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 709
- ULFileTransfer constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 247
- ULFileTransfer constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 711
- ULFileTransfer members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 709
- ULFileTransferProgressData class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 260
- ULFileTransferProgressData class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 724
- ULFileTransferProgressData members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 724
- ULFileTransferProgressListener interface [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 263
- ULFileTransferProgressListener interface [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 727
- ULFileTransferProgressListener members [UL .NET 2.0]

- iAnywhere.Data.UltraLite namespace, 727
- ULIndexSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 265
- ULIndexSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 729
- ULIndexSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 729
- ULInfoMessageEventArgs class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 273
- ULInfoMessageEventArgs class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 738
- ULInfoMessageEventArgs members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 738
- ULInfoMessageEventHandler delegate [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 276
- ULInfoMessageEventHandler delegate [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 741
- ULMetaDataCollectionNames class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 742
- ULMetaDataCollectionNames members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 742
- ULParameter class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 277
- ULParameter class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 751
- ULParameter constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 279, 280, 281, 282, 283
- ULParameter constructors [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 753
- ULParameter members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 752
- ULParameter() constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 753
- ULParameter(String, Object) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 754
- ULParameter(String, ULDbType) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 755
- ULParameter(String, ULDbType, Int32) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 755
- ULParameter(String, ULDbType, Int32, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 757
- ULParameter(String, ULDbType, Int32, String) constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 756
- ULParameterCollection class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 294
- ULParameterCollection class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 768
- ULParameterCollection members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 768
- ULPublicationSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 312
- ULPublicationSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 788
- ULPublicationSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 789
- ULResultSet class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 316
- ULResultSet class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 792
- ULResultSet members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 793
- ULResultSetSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 342
- ULResultSetSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 816
- ULResultSetSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 816
- ULRowsCopied event [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 479
- ULRowsCopiedEventArgs class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 819
- ULRowsCopiedEventArgs constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 819
- ULRowsCopiedEventArgs members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 819
- ULRowsCopiedEventHandler delegate [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 822
- ULRowUpdatedEventArgs class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 344
- ULRowUpdatedEventArgs class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 823
- ULRowUpdatedEventArgs constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 345
- ULRowUpdatedEventArgs constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 824
- ULRowUpdatedEventArgs members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 823
- ULRowUpdatedEventHandler delegate [UL .NET 1.0]

- iAnywhere.Data.UltraLite namespace, 348
- ULRowUpdatedEventHandler delegate [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 827
- ULRowUpdatingEventArgs class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 349
- ULRowUpdatingEventArgs class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 828
- ULRowUpdatingEventArgs constructor [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 350
- ULRowUpdatingEventArgs constructor [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 829
- ULRowUpdatingEventArgs members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 828
- ULRowUpdatingEventHandler delegate [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 352
- ULRowUpdatingEventHandler delegate [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 831
- ULRuntimeType enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 353
- ULRuntimeType enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 832
- ULServerSyncListener interface [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 354
- ULServerSyncListener interface [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 833
- ULServerSyncListener members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 833
- ULSQLCode enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 357
- ULSQLCode enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 836
- ULStreamErrorCode enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 366
- ULStreamErrorCode enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 846
- ULStreamErrorContext enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 381
- ULStreamErrorContext enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 864
- ULStreamErrorID enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 382
- ULStreamErrorID enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 865
- ULStreamType enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 384
- ULStreamType enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 867
- ULSyncParms class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 385
- ULSyncParms class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 868
- ULSyncParms members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 868
- ULSyncProgressData class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 399
- ULSyncProgressData class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 883
- ULSyncProgressData members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 883
- ULSyncProgressListener interface [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 409
- ULSyncProgressListener interface [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 894
- ULSyncProgressListener members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 894
- ULSyncProgressState enumeration [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 411
- ULSyncProgressState enumeration [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 896
- ULSyncResult class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 413
- ULSyncResult class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 898
- ULSyncResult members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 898
- ULTable class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 419
- ULTable class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 904
- ULTable members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 904
- ULTableSchema class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 441
- ULTableSchema class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 925
- ULTableSchema members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 925
- UltraLite
 - Visual Studio integration, 36
- UltraLite databases
 - UltraLite.NET connections, 15
 - UltraLite.NET information access , 29
- UltraLite modes
 - UltraLite.NET, 23
- UltraLite.NET
 - about, 1

- accessing schema information, 29
 - activesync synchronization , 33
 - architecture, 5
 - benefits, 2
 - data manipulation, 18
 - data manipulation with SQL, 18
 - data retrieval, 20
 - deploying, 48
 - development, 13
 - dynamic SQL tutorial, 43
 - encryption, 17
 - error handling, 30
 - SQL Anywhere Explorer for UltraLite, 7
 - supported platforms, 4
 - synchronization example , 32
 - synchronization in applications, 32
 - Table API introduction, 22
 - transaction processing, 28
 - tutorials, 35
 - user authentication, 31
 - ULTransaction class [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 456
 - ULTransaction class [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 939
 - ULTransaction members [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 939
 - understanding UltraLite.Net development
 - about, 13
 - UnusedEvent event [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 132
 - UnusedEvent event [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 588
 - Update method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 172, 173, 174, 175, 341
 - Update method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 815
 - update mode
 - UltraLite.NET, 23
 - UpdateBegin method [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 439
 - UpdateBegin method [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 923
 - UpdateCommand property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 163
 - UpdateCommand property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 635
 - UpdatedRowSource property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 69
 - UpdatedRowSource property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 509
 - updating
 - UltraLite.NET table rows, 25
 - UploadOK property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 418
 - UploadOK property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 903
 - UploadOnly property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 396
 - UploadOnly property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 880
 - UploadUnchangedRows property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 445
 - UploadUnchangedRows property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 929
 - user authentication
 - UltraLite.NET development, 31
 - UserID property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 131
 - UserID property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 587, 602
 - UserName property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 256, 397
 - UserName property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 720, 881
 - using the SQL Anywhere Explorer for UltraLite
 - about, 9
 - using UltraLite modes
 - UltraLite.NET, 23
 - UTF8Encoding property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 142
 - UTF8Encoding property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 616
- ## V
- Value property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 292
 - Value property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 765
 - values
 - UltraLite.NET accessing in , 23
 - Version property [UL .NET 1.0]
 - iAnywhere.Data.UltraLite namespace, 257, 397
 - Version property [UL .NET 2.0]
 - iAnywhere.Data.UltraLite namespace, 720, 881

Visual Studio
 integration with UltraLite, 36
 SQL Anywhere Explorer integration with UltraLite,
 9
Visual Studio .NET
 accessing UltraLite databases, 9
 UltraLite database connections, 9
 UltraLite.NET connecting to UltraLite databases,
 14

W

Windows CE
 UltraLite.NET target platforms, 4
working with UltraLite tables using the SQL Anywhere
Explorer
 about, 11
WriteToServer methods [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 476
WriteToServer(DataRow[]) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 476
WriteToServer(DataTable) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 477
WriteToServer(DataTable, DataRowState) method
[UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 478
WriteToServer(IDataReader) method [UL .NET 2.0]
 iAnywhere.Data.UltraLite namespace, 477
