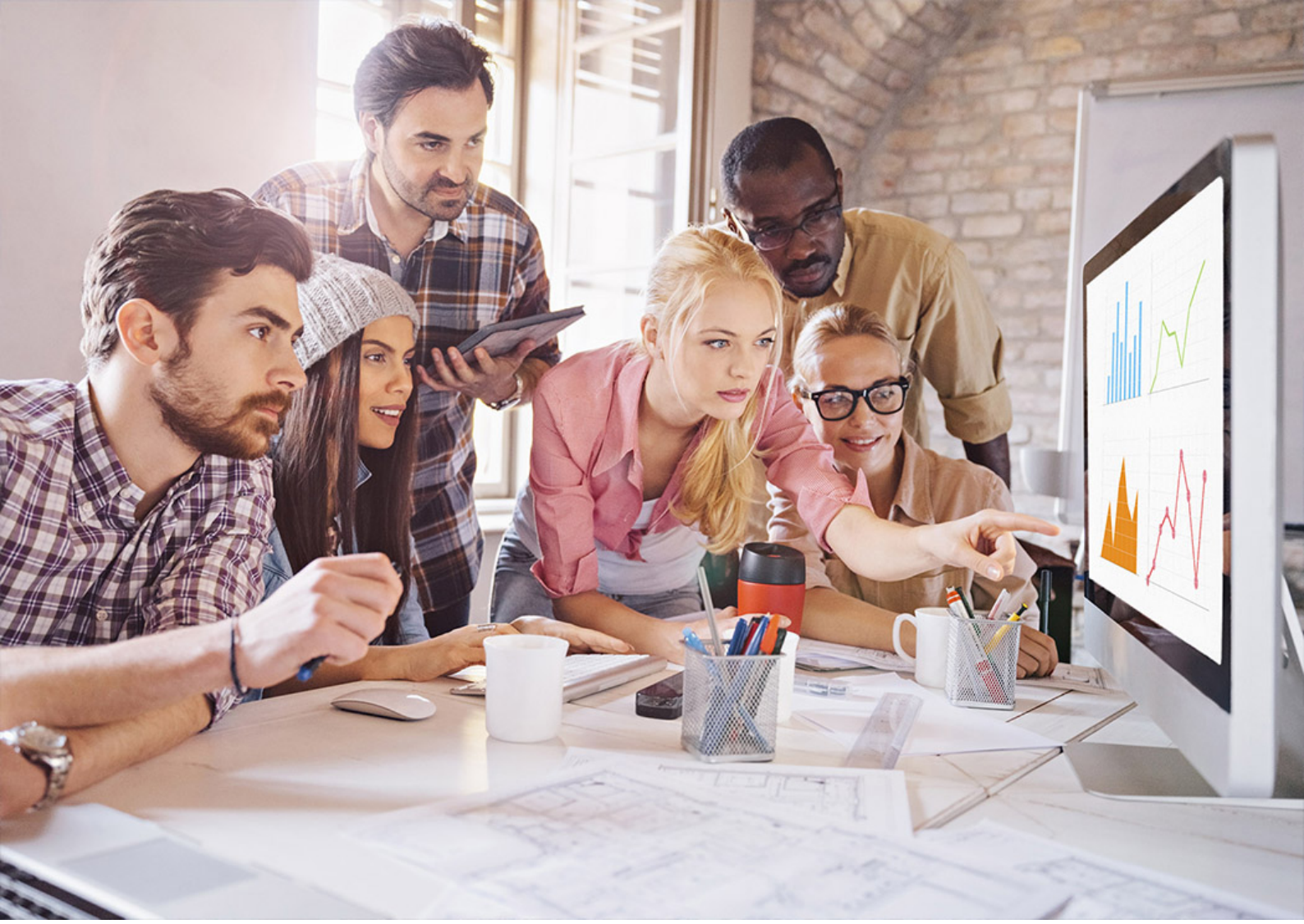

PUBLIC
SQL Anywhere - UltraLite
Document Version: 17.01.0 – 2021-10-15

UltraLite - .NET API Reference

THE BEST RUN SAP

Content

1 UltraLite .NET API Reference

You can write .NET applications for UltraLite by using the API for the UltraLite.NET Data Provider for .NET Framework 2.0 and .NET Compact Framework 2.0.

The following list describes some of the more commonly-used high level classes for the Sap.Data.UltraLite 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.

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.

In this section:

UltraLite .NET API [page 4]
Use the API for the UltraLite.NET Data Provider for .NET Framework 2.0 and .NET Compact Framework 2.0 to write .NET applications for UltraLite.

1.1 UltraLite .NET API

Use the API for the UltraLite.NET Data Provider for .NET Framework 2.0 and .NET Compact Framework 2.0 to write .NET applications for UltraLite.

Namespace

```
Sap.Data.UltraLite
```

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 the RuntimeType property to the appropriate value before using any other UltraLite.NET API.

The assembly uses a satellite resource assembly named Sap.Data.UltraLite.resources. The main assembly searches for this resource assembly by culture, using the following order:

- CultureInfo.CurrentUICulture
- CultureInfo.CurrentCulture
- EN

In this section:

UL Ext: Represents the schema of an UltraLite.NET database.

ULDataReader Class [page 260]
Represents a read-only bi-directional cursor in an UltraLite database.

ULException Class [page 312]
Represents a SQL error returned by the UltraLite.NET database.

ULFactory Class [page 314]
Represents a set of methods for creating instances of the Sap.Data.UltraLite provider's implementation of the data source classes.

ULFileTransfer Class [page 320]
UL Ext: Transfers a file from a remote database using the MobiLink server.

ULFileTransferProgressData Class [page 341]
UL Ext: Returns file transfer progress monitoring data.

ULFileTransferProgressListener Interface [page 344]
UL Ext: The listener interface for receiving file transfer progress events.

ULIndexSchema Class [page 346]
UL Ext: Represents the schema of an UltraLite table index.

ULInfoMessageEventArgs Class [page 356]
Provides data for the ULConnection.InfoMessage event.

ULMetaDataCollectionNames Class [page 359]
Provides a list of constants for use with the ULConnection.GetSchema(String,String[]) method to retrieve metadata collections.

ULParameter Class [page 372]
Represents a parameter to a ULCommand object.

ULParameterCollection Class [page 392]
Represents all parameters to a ULCommand object.

ULResultSet Class [page 421]
UL Ext: Represents an editable result set in an UltraLite database.

ULResultSetSchema Class [page 453]
UL Ext: Represents the schema of an UltraLite result set.

ULRowsCopiedEventArgs Class [page 455]
Represents the set of arguments passed to the ULRowsCopiedEventHandler object.

ULRowUpdatedEventArgs Class [page 458]
Provides data for the ULDataAdapter.RowUpdated event.

ULRowUpdatingEventArgs Class [page 461]
Provides data for the ULDataAdapter.RowUpdating event.

ULServerSyncListener Interface [page 464]
UL Ext: The listener interface for receiving server synchronization messages.

ULSqlProgressData Class (Deprecated) [page 466]
UL Ext: Returns SQL passthrough script progress monitoring data.

ULSyncParms Class [page 468]
UL Ext: Represents synchronization parameters that define how to synchronize an UltraLite database.

ULSyncProgressData Class [page 484]

UL Ext: Returns synchronization progress monitoring data.

ULSyncProgressListener Interface [page 499]
UL Ext: The listener interface for receiving synchronization progress events.

ULSyncResult Class [page 501]
UL Ext: Represents the status of the last synchronization.

ULTable Class [page 507]
UL Ext: Represents a table in an UltraLite database.

ULTableSchema Class [page 537]
UL Ext: Represents the schema of an UltraLite table.

ULTransaction Class [page 557]
Represents a SQL transaction.

ULInfoMessageEventHandler(object, ULInfoMessageEventArgs) Delegate [page 562]
Represents the method that handles the ULConnection.InfoMessage event.

ULRowsCopiedEventHandler(object, ULRowsCopiedEventArgs) Delegate [page 563]
Represents the method that handles the ULBulkCopy.ULRowsCopied event.

ULRowUpdatedEventHandler(object, ULRowUpdatedEventArgs) Delegate [page 563]
Represents the method that handles the ULDataAdapter.RowUpdated event.

ULRowUpdatingEventHandler(object, ULRowUpdatingEventArgs) Delegate [page 564]
Represents the method that handles the ULDataAdapter.RowUpdating event.

ULSyncProgressedDlg(IAsyncResult, ULSyncProgressData) Delegate [page 565]
Represents the method that is invoked during synchronization with synchronization progress information.

ULAuthStatusCode Enumeration [page 566]
UL Ext: Enumerates the status codes that may be reported during MobiLink user authentication.

ULBulkCopyOptions Enumeration [page 567]
A bitwise flag that specifies one or more options to use with an instance of the ULBulkCopy class.

ULDateOrder Enumeration [page 568]
UL Ext: Enumerates the date orders that a database can support.

ULDbType Enumeration [page 568]
Enumerates the UltraLite.NET database data types.

ULDBValid Enumeration [page 571]
Enumerates the UltraLite.NET database validation methods

ULRuntimeType Enumeration [page 572]
UL Ext: Enumerates the types of UltraLite.NET runtimes.

ULSqlProgressState Enumeration [page 573]
UL Ext: Enumerates all the states that can occur while executing SQL passthrough scripts.

ULStreamType Enumeration [page 574]
UL Ext: Enumerates the types of MobiLink synchronization streams to use for synchronization.

ULSyncProgressState Enumeration [page 575]
UL Ext: Enumerates all the states that can occur while synchronizing.

1.1.1 ULActiveSyncListener Interface

UL Ext: The listener interface for receiving ActiveSync events.

Syntax

Visual Basic

```
Public Interface ULActiveSyncListener
```

C#

```
public interface ULActiveSyncListener
```

Members

All members of ULActiveSyncListener, including inherited members.

Methods

Modifier and Type	Method	Description
public void	ActiveSyncInvoked(bool) [page 8]	Invoked when the MobiLink provider for ActiveSync calls the application to perform synchronization.

In this section:

1.1.1.1 ActiveSyncInvoked(bool) 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, 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 the ULDatabaseManager.SignalSyncIsComplete method 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 Sap.Data.UltraLite
Public Class MainWindow
   Inherits System.Windows.Forms.Form
   Implements ULActiveSyncListener
   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 ULActiveSyncListener.ActiveSyncInvoked
       Me.Invoke(New EventHandler(AddressOf Me.ActiveSyncAction))
   End Sub
   Public Sub ActiveSyncAction( _
       ByVal sender As Object, ByVal e As EventArgs _
   )
       ' Perform active sync.
```

```
        conn.Synchronize()
        ULConnection.DatabaseManager.SignalSyncIsComplete()
    End Sub
End Class
```

The following code is the C# language equivalent:

```
// C#
using Sap.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 the
       // InitializeComponent call.
       //
       ULDatabaseManager.SetActiveSyncListener(
           "myCompany.myapp", this
       );
       // Create connection
       ...
   }
   protected override void Dispose( bool disposing )
   {
       base.Dispose( disposing );
   }
   protected override void OnClosing(
       System.ComponentModel.CancelEventArgs e )
    {
       ULDatabaseManager.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();
       ULDatabaseManager.SignalSyncIsComplete();
   }
}
```

Related Information

SignalSyncIsComplete() Method [page 247]

1.1.2 ULBulkCopy Class

Efficiently bulk loads an UltraLite table with data from another source.

Syntax

Visual Basic

```
Public NotInheritable Class ULBulkCopy Implements System.IDisposable
```

C#

```
public sealed class ULBulkCopy : System.IDisposable
```

Members

All members of ULBulkCopy, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULBulkCopy [page 13]	Initializes a ULBulkCopy object with the specified ULConnection object.

Methods

Modifier and Type	Method	Description
public void	Close() [page 17]	Closes the ULBulkCopy object.
public void	Dispose() [page 17]	Disposes of the ULBulkCopy object.
public void	WriteToServer [page 18]	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.

Properties

Modifier and Type	Property	Description
public int	BatchSize [page 21]	Gets or sets the number of rows in each batch.
public int	BulkCopyTimeout [page 22]	Gets or sets the number of seconds for the operation to complete before it times out.
public ULBulkCopyColumnMappingCollection	ColumnMappings [page 22]	Returns a collection of ULBulkCopyColumnMapping items.
public string	DestinationTableName [page 23]	Gets or sets the name of the destination table on the server.

Modifier and Type	Property	Description
public int	NotifyAfter [page 24]	Specifies the number of rows to be processed before generating a notification event.

Events

Modifier and Type	Event	Description
public ULRowsCopiedEventHandler	ULRowsCopied [page 24]	This event occurs every time the number of rows specified by the NotifyAfter property have been processed.

Remarks

The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

In this section:

ULBulkCopy Constructor [page 13]
Initializes a ULBulkCopy object with the specified ULConnection object.

Close() Method [page 17]
Closes the ULBulkCopy object.

Dispose() Method [page 17]
Disposes of the ULBulkCopy object.

WriteToServer Method [page 18]
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.

BatchSize Property [page 21]
Gets or sets the number of rows in each batch.

BulkCopyTimeout Property [page 22]
Gets or sets the number of seconds for the operation to complete before it times out.

ColumnMappings Property [page 22]
Returns a collection of ULBulkCopyColumnMapping items.

DestinationTableName Property [page 23]
Gets or sets the name of the destination table on the server.

NotifyAfter Property [page 24]
Specifies the number of rows to be processed before generating a notification event.

ULRowsCopied Event [page 24]
This event occurs every time the number of rows specified by the NotifyAfter property have been processed.

1.1.2.1 ULBulkCopy Constructor

Initializes a ULBulkCopy object with the specified ULConnection object.

Overload List

Modifier and Type	Overload name	Description
public	ULBulkCopy(ULConnection) [page 13]	Initializes a ULBulkCopy object with the specified ULConnection object.
public	ULBulkCopy(ULConnection, ULBulkCopyOptions, ULTransaction) [page 14]	Initializes a ULBulkCopy object with the specified ULConnection object, copy options, and ULTransaction object.
public	ULBulkCopy(string) [page 15]	Initializes a ULBulkCopy object with the specified connection string.
public	ULBulkCopy(string, ULBulkCopyOptions) [page 16]	Initializes a ULBulkCopy object with the specified connection string, and copy options.

In this section:

ULBulkCopy(ULConnection) Constructor [page 13]
Initializes a ULBulkCopy object with the specified ULConnection object.

ULBulkCopy(ULConnection, ULBulkCopyOptions, ULTransaction) Constructor [page 14]
Initializes a ULBulkCopy object with the specified ULConnection object, copy options, and ULTransaction object.

ULBulkCopy(string) Constructor [page 15]
Initializes a ULBulkCopy object with the specified connection string.

ULBulkCopy(string, ULBulkCopyOptions) Constructor [page 16]
Initializes a ULBulkCopy object with the specified connection string, and copy options.

1.1.2.1.1 ULBulkCopy(ULConnection) Constructor

Initializes a ULBulkCopy object with the specified ULConnection object.

Syntax

Visual Basic

```
Public Sub ULBulkCopy (ByVal connection As ULConnection)
```

C#

```
public ULBulkCopy (ULConnection connection)
```

Parameters

connection The ULConnection object that is used to perform the bulk-copy operation. If the connection is not open, an exception is thrown during a WriteToServer method call.

Remarks

The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

Related Information

ULConnection Class [page 108]

1.1.2.1.2 ULBulkCopy(ULConnection, ULBulkCopyOptions, ULTransaction) Constructor

Initializes a ULBulkCopy object with the specified ULConnection object, copy options, and ULTransaction object.

Syntax

Visual Basic

```
Public Sub ULBulkCopy (
    ByVal connection As ULConnection,
    ByVal copyOptions As ULBulkCopyOptions,
    ByVal externalTransaction As ULTransaction
)
```

C#

```
public ULBulkCopy (
    ULConnection connection,
    ULBulkCopyOptions copyOptions,
    ULTransaction externalTransaction
)
```

Parameters

connection The ULConnection object that is used to perform the bulk-copy operation. If the connection is not open, an exception is thrown during a WriteToServer method call.

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 object under which the bulk copy occurs. If this value 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

The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

Related Information

ULConnection Class [page 108]
ULTransaction Class [page 557]

1.1.2.1.3 ULBulkCopy(string) Constructor

Initializes a ULBulkCopy object with the specified connection string.

Syntax

Visual Basic

```
Public Sub ULBulkCopy (ByVal connectionString As String)
```

C#

```
public ULBulkCopy (string connectionString)
```

Parameters

connectionString The string defining the connection to be opened for use by the ULBulkCopy object. A connection string is a semicolon-separated list of keyword=value pairs.

Remarks

The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

This syntax opens a connection during a WriteToServer method call with the connectionString value. The connection is closed at the end of the WriteToServer call.

The connection string can be supplied using a ULConnectionParms object.

Related Information

ConnectionString Property [page 163]
ULConnectionParms Class [page 173]

1.1.2.1.4 ULBulkCopy(string, ULBulkCopyOptions) Constructor

Initializes a ULBulkCopy object with the specified connection string, and copy options.

Syntax

Visual Basic

```
Public Sub ULBulkCopy (
    ByVal connectionString As String,
    ByVal copyOptions As ULBulkCopyOptions
)
```

C#

```
public ULBulkCopy (
    string connectionString,
    ULBulkCopyOptions copyOptions
)
```

Parameters

connectionString The string defining the connection to be opened for use by the ULBulkCopy object. A connection string is a semicolon-separated list of keyword=value pairs.

copyOptions A combination of values from the ULBulkCopyOptions enumeration that determines how data source rows are copied to the destination table.

Remarks

The ULBulkCopy class is not available in the .NET Compact Framework 2.0.

This syntax opens a connection during a WriteToServer method call with the connectionString value. The connection is closed at the end of the WriteToServer call.

Related Information

ConnectionString Property [page 163]
ULBulkCopyOptions Enumeration [page 567]

1.1.2.2 Close() Method

Closes the ULBulkCopy object.

Syntax

Visual Basic

```
Public Sub Close ()
```

C#

```
public void Close ()
```

1.1.2.3 Dispose() Method

Disposes of the ULBulkCopy object.

Syntax

Visual Basic

```
Public Sub Dispose ()
```

C#

```
public void Dispose ()
```

1.1.2.4 WriteToServer 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.

Overload List

Modifier and Type	Overload name	Description
public void	WriteToServer(DataRow[]) [page 19]	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 void	WriteToServer(DataTable) [page 19]	Copies all rows in the supplied System.Data.DataTable to a destination table specified by the ULBulkCopy.DestinationTableName property.
public void	WriteToServer(DataTable, DataRowState) [page 20]	Copies all rows in the supplied System.Data.DataTable with the specified row state to a destination table specified by the ULBulkCopy.DestinationTableName property.
public void	WriteToServer(IDataReader) [page 21]	Copies all rows in the supplied System.Data.IDataReader to a destination table specified by the ULBulkCopy.DestinationTableName property.

In this section:

WriteToServer(DataRow[]) Method [page 19]
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(DataTable) Method [page 19]
Copies all rows in the supplied System.Data.DataTable to a destination table specified by the ULBulkCopy.DestinationTableName property.

WriteToServer(DataTable, DataRowState) Method [page 20]
Copies all rows in the supplied System.Data.DataTable with the specified row state to a destination table specified by the ULBulkCopy.DestinationTableName property.

WriteToServer(IDataReader) Method [page 21]
Copies all rows in the supplied System.Data.IDataReader to a destination table specified by the ULBulkCopy.DestinationTableName property.

1.1.2.4.1 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 to be copied to the destination table.

Related Information

DestinationTableName Property [page 23]

1.1.2.4.2 WriteToServer(DataTable) Method

Copies all rows in the supplied System.Data.DataTable to a destination table specified by the ULBulkCopy.DestinationTableName property.

Syntax

Visual Basic

```
Public Sub WriteToServer (ByVal table As DataTable)
```

C#

```
public void WriteToServer (DataTable table)
```

Parameters

table A System.Data.DataTable whose rows to be copied to the destination table.

Related Information

DestinationTableName Property [page 23]

1.1.2.4.3 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 ULBulkCopy.DestinationTableName property.

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 to 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 the rowState parameter is specified, then only those rows that have the same row state are copied.

Related Information

DestinationTableName Property [page 23]

1.1.2.4.4 WriteToServer(IDataReader) Method

Copies all rows in the supplied System.Data.IDataReader to a destination table specified by the ULBulkCopy.DestinationTableName property.

Syntax

Visual Basic

```
Public Sub WriteToServer (ByVal reader As IDataReader)
```

C#

```
public void WriteToServer (IDataReader reader)
```

Parameters

reader A System.Data.IDataReader whose rows to be copied to the destination table.

Related Information

DestinationTableName Property [page 23]

1.1.2.5 BatchSize Property

Gets or sets the number of rows in each batch.

Syntax

Visual Basic

```
Public Property BatchSize As Integer
```

C#

```
public int BatchSize {get;set;}
```

Remarks

At the end of each batch, the rows in the batch are sent to the server.

The number of rows in each batch. The default is 0.

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.

1.1.2.6 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;}
```

Remarks

The default value is 30 seconds.

Setting a value of zero indicates no limit, which 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 error is thrown.

Setting a value that is less than zero would throw an error.

1.1.2.7 ColumnMappings Property

Returns a collection of ULBulkCopyColumnMapping items.

Syntax

Visual Basic

```
Public ReadOnly Property ColumnMappings As
ULBulkCopyColumnMappingCollection
```

C#

```
public ULBulkCopyColumnMappingCollection ColumnMappings {get;}
```

Remarks

Column mappings define the relationships between columns in the data source and columns in the destination.

By default, it is an empty collection.

The property cannot be modified while a WriteToServer method call is executing.

If the ColumnMappings object is empty when the WriteToServer method is executed, then the first column in the source is mapped to the first column in the destination, the second is mapped 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.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.2.8 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;}
```

Remarks

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

If the value is changed while a WriteToServer call is executing, the change has no effect.

If the value has not been set before a call to the WriteToServer method, an InvalidOperationException error is thrown.

It is an error to set the value to null (Nothing in Visual Basic) or the empty string.

1.1.2.9 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;}
```

Remarks

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.

Changes made to this property, while executing the WriteToServer method, do not take effect until after the next notification.

Setting this value ta a value that is less than zero throws an error.

The values of NotifyAfter and BulkCopyTimeout properties are mutually exclusive, so the event can fire even if no rows have been sent to the database or committed.

Related Information

BulkCopyTimeout Property [page 22]

1.1.2.10 ULRowsCopied Event

This event occurs every time the number of rows specified by the NotifyAfter property have been processed.

Syntax

Visual Basic

```
Public Event ULRowsCopied  As ULRowsCopiedEventHandler
```

C#

```
public 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.

Related Information

NotifyAfter Property [page 24]

1.1.3 ULBulkCopyColumnMapping Class

Defines the mapping between a column in a ULBulkCopy instance's data source and a column in the instance's destination table.

Syntax

Visual Basic

```
Public NotInheritable Class ULBulkCopyColumnMapping
```

C#

```
public sealed class ULBulkCopyColumnMapping
```

Members

All members of ULBulkCopyColumnMapping, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULBulkCopyColumnMapping [page 26]	Creates a new column mapping.

Properties

Modifier and Type	Property	Description
public string	DestinationColumn [page 31]	Specifies the name of the column in the destination database table being mapped to.
public int	DestinationOrdinal [page 32]	Specifies the ordinal value of the column in the destination database table being mapped to.

Modifier and Type	Property	Description
public string	SourceColumn [page 32]	Specifies the name of the column being mapped in the data source.
public int	SourceOrdinal [page 33]	Specifies the ordinal position of the source column within the data source.

Remarks

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

In this section:

ULBulkCopyColumnMapping Constructor [page 26]
Creates a new column mapping.

DestinationColumn Property [page 31]
Specifies the name of the column in the destination database table being mapped to.

DestinationOrdinal Property [page 32]
Specifies the ordinal value of the column in the destination database table being mapped to.

SourceColumn Property [page 32]
Specifies the name of the column being mapped in the data source.

SourceOrdinal Property [page 33]
Specifies the ordinal position of the source column within the data source.

Related Information

ULBulkCopy Class [page 11]

1.1.3.1 ULBulkCopyColumnMapping Constructor

Creates a new column mapping.

Overload List

Modifier and Type	Overload name	Description
public	ULBulkCopyColumnMapping() [page 27]	Creates a new column mapping.

Modifier and Type	Overload name	Description
public	ULBulkCopyColumnMapping(int, int) [page 28]	Creates a new column mapping, using column ordinals or names to refer to source and destination columns.
public	ULBulkCopyColumnMapping(int, string) [page 29]	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.
public	ULBulkCopyColumnMapping(string, int) [page 29]	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.
public	ULBulkCopyColumnMapping(string, string) [page 30]	Creates a new column mapping, using column names to refer to source and destination columns.

In this section:

ULBulkCopyColumnMapping() Constructor [page 27]
Creates a new column mapping.

ULBulkCopyColumnMapping(int, int) Constructor [page 28]
Creates a new column mapping, using column ordinals or names to refer to source and destination columns.

ULBulkCopyColumnMapping(int, string) Constructor [page 29]
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.

ULBulkCopyColumnMapping(string, int) Constructor [page 29]
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.

ULBulkCopyColumnMapping(string, string) Constructor [page 30]
Creates a new column mapping, using column names to refer to source and destination columns.

1.1.3.1.1 ULBulkCopyColumnMapping() Constructor

Creates a new column mapping.

Syntax

Visual Basic

```
Public Sub ULBulkCopyColumnMapping ()
```

C#

```
public ULBulkCopyColumnMapping ()
```

Remarks

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

1.1.3.1.2 ULBulkCopyColumnMapping(int, int) Constructor

Creates a new column mapping, using column ordinals or names to refer to source and destination columns.

Syntax

Visual Basic

```
Public Sub ULBulkCopyColumnMapping (
    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

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

1.1.3.1.3 ULBulkCopyColumnMapping(int, 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 ULBulkCopyColumnMapping (
    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

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

1.1.3.1.4 ULBulkCopyColumnMapping(string, int) 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 ULBulkCopyColumnMapping (
    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

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

1.1.3.1.5 ULBulkCopyColumnMapping(string, string) Constructor

Creates a new column mapping, using column names to refer to source and destination columns.

Syntax

Visual Basic

```
Public Sub ULBulkCopyColumnMapping (
    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

The ULBulkCopyColumnMapping class is not available in the .NET Compact Framework 2.0.

1.1.3.2 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;}
```

Remarks

A string specifying the name of the column in the destination table or a null reference (Nothing in Visual Basic) if the DestinationOrdinal property has priority.

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 the DestinationColumn property to null or the empty string.

Related Information

DestinationOrdinal Property [page 32]
DestinationOrdinal Property [page 32]

1.1.3.3 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;}
```

Remarks

An integer specifying the ordinal of the column being mapped to in the destination table or -1 if the property is not set.

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).

Related Information

DestinationColumn Property [page 31]
DestinationColumn Property [page 31]

1.1.3.4 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;}
```

Remarks

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.

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 -1. 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 the SourceColumn property to null or the empty string.

Related Information

SourceOrdinal Property [page 33]
SourceOrdinal Property [page 33]

1.1.3.5 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;}
```

Remarks

An integer specifying the ordinal of the column in the data source or -1 if the property is not set.

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 -1. Setting the SourceOrdinal property causes the SourceColumn property to be set to a null reference (Nothing in Visual Basic).

Related Information

SourceColumn Property [page 32]

1.1.4 ULBulkCopyColumnMappingCollection Class

A collection of ULBulkCopyColumnMapping objects that inherits from System.Collections.CollectionBase.

Syntax

Visual Basic

```
Public NotInheritable Class ULBulkCopyColumnMappingCollection Inherits
System.Collections.CollectionBase
```

C#

```
public sealed class ULBulkCopyColumnMappingCollection :
System.Collections.CollectionBase
```

Members

All members of ULBulkCopyColumnMappingCollection, including inherited members.

Methods

Modifier and Type	Method	Description
public ULBulkCopyColumnMapping	Add [page 36]	Adds the specified ULBulkCopyColumnMapping object to the collection.
public bool	Contains(ULBulkCopyColumnMapping) [page 42]	Returns whether the specified ULBulkCopyColumnMapping object exists in the collection.
public void	CopyTo(ULBulkCopyColumnMapping[], int) [page 42]	Copies the elements of the ULBulkCopyColumnMappingCollection object to an array of ULBulkCopyColumnMapping objects, starting at a particular index.
public int	IndexOf(ULBulkCopyColumnMapping) [page 43]	Returns the index of the specified ULBulkCopyColumnMapping object within the collection.
public void	Remove(ULBulkCopyColumnMapping) [page 44]	Removes the specified ULBulkCopyColumnMapping object from the ULBulkCopyColumnMappingCollection object.
public new void	RemoveAt(int) [page 44]	Removes the mapping at the specified index from the collection.

Properties

Modifier and Type	Property	Description
public ULBulkCopyColumnMapping	this[int index] [page 45]	Gets the ULBulkCopyColumnMapping object at the specified index.

Remarks

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

In this section:

Add Method [page 36]
Adds the specified ULBulkCopyColumnMapping object to the collection.

Contains(ULBulkCopyColumnMapping) Method [page 42]
Returns whether the specified ULBulkCopyColumnMapping object exists in the collection.

CopyTo(ULBulkCopyColumnMapping[], int) Method [page 42]
Copies the elements of the ULBulkCopyColumnMappingCollection object to an array of ULBulkCopyColumnMapping objects, starting at a particular index.

IndexOf(ULBulkCopyColumnMapping) Method [page 43]
Returns the index of the specified ULBulkCopyColumnMapping object within the collection.

Remove(ULBulkCopyColumnMapping) Method [page 44]
Removes the specified ULBulkCopyColumnMapping object from the ULBulkCopyColumnMappingCollection object.

RemoveAt(int) Method [page 44]
Removes the mapping at the specified index from the collection.

this[int index] Property [page 45]
Gets the ULBulkCopyColumnMapping object at the specified index.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.1 Add Method

Adds the specified ULBulkCopyColumnMapping object to the collection.

Overload List

Modifier and Type	Overload name	Description
public ULBulkCopyColumnMapping	Add(ULBulkCopyColumnMapping) [page 37]	Adds the specified ULBulkCopyColumnMapping object to the collection.
public ULBulkCopyColumnMapping	Add(int, int) [page 38]	Creates a new ULBulkCopyColumnMapping object using ordinals to specify both source and destination columns, and adds the mapping to the collection.
public ULBulkCopyColumnMapping	Add(int, string) [page 39]	Creates a new ULBulkCopyColumnMapping object 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.
public ULBulkCopyColumnMapping	Add(string, int) [page 40]	Creates a new ULBulkCopyColumnMapping object 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.
public ULBulkCopyColumnMapping	Add(string, string) [page 41]	Creates a new ULBulkCopyColumnMapping object using column names to specify both source and destination columns, and adds the mapping to the collection.

In this section:

Add(ULBulkCopyColumnMapping) Method [page 37]
Adds the specified ULBulkCopyColumnMapping object to the collection.

Add(int, int) Method [page 38]
Creates a new ULBulkCopyColumnMapping object using ordinals to specify both source and destination columns, and adds the mapping to the collection.

Add(int, string) Method [page 39]
Creates a new ULBulkCopyColumnMapping object 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.

Add(string, int) Method [page 40]
Creates a new ULBulkCopyColumnMapping object 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.

Add(string, string) Method [page 41]

Creates a new ULBulkCopyColumnMapping object using column names to specify both source and destination columns, and adds the mapping to the collection.

1.1.4.1.1 Add(ULBulkCopyColumnMapping) Method

Adds the specified ULBulkCopyColumnMapping object 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

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.1.2 Add(int, int) Method

Creates a new ULBulkCopyColumnMapping object 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

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.1.3 Add(int, string) Method

Creates a new ULBulkCopyColumnMapping object 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

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.1.4 Add(string, int) Method

Creates a new ULBulkCopyColumnMapping object 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.

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

Creates a new column mapping, using column ordinals or names to refer to source and destination columns.

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.1.5 Add(string, string) Method

Creates a new ULBulkCopyColumnMapping object 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

The ULBulkCopyColumnMappingCollection class is not available in the .NET Compact Framework 2.0.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.2 Contains(ULBulkCopyColumnMapping) 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.

Returns

True if the specified mapping exists in the collection; otherwise, returns false.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.3 CopyTo(ULBulkCopyColumnMapping[], int) Method

Copies the elements of the ULBulkCopyColumnMappingCollection object to an array of ULBulkCopyColumnMapping objects, 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 object. The array must have zero-based indexing.
index The zero-based index in the array at which copying begins.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.4 IndexOf(ULBulkCopyColumnMapping) Method

Returns the index of the specified ULBulkCopyColumnMapping object 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.

Returns

The zero-based index of the column mapping is returned, or -1 is returned if the column mapping is not found in the collection.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.5 Remove(ULBulkCopyColumnMapping) Method

Removes the specified ULBulkCopyColumnMapping object from the ULBulkCopyColumnMappingCollection object.

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.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.4.6 RemoveAt(int) Method

Removes the mapping at the specified index from the collection.

Syntax

Visual Basic

```
Public Shadows Sub RemoveAt (ByVal index As Integer)
```

C#

```
public new void RemoveAt (int index)
```

Parameters

index The zero-based index of the ULBulkCopyColumnMapping object to be removed from the collection.

1.1.4.7 this[int index] 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;}
```

Remarks

An ULBulkCopyColumnMapping object is returned.

Related Information

ULBulkCopyColumnMapping Class [page 25]

1.1.5 ULCommand Class

Represents a pre-compiled SQL statement or query, with or without IN parameters.

Syntax

Visual Basic

```
Public NotInheritable Class ULCommand Inherits
System.Data.Common.DbCommand Implements System.ICloneable
```

C#

```
public sealed class ULCommand : System.Data.Common.DbCommand,
System.ICloneable
```

Members

All members of ULCommand, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULCommand [page 50]	Initializes a ULCommand object.

Methods

Modifier and Type	Method	Description
public IAsyncResult	BeginExecuteNonQuery [page 55]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object.
public IAsyncResult	BeginExecuteReader [page 57]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set.
public override void	Cancel() [page 62]	This method is not supported in UltraLite.NET.
protected override DbParameter	CreateDbParameter() [page 62]	
public new ULParameter	CreateParameter() [page 63]	Provides a ULParameter object for supplying parameters to ULCommand objects.
protected override void	Dispose(bool) [page 63]	Releases the unmanaged resources used by the ULCommand object and optionally releases the managed resources.
public int	EndExecuteNonQuery(IAsyncResult) [page 64]	Finishes asynchronous execution of a SQL statement.
public ULDataReader	EndExecuteReader(IAsyncResult) [page 68]	Finishes asynchronous execution of a SQL statement, returning the requested ULDataReader.
protected override DbDataReader	ExecuteDbDataReader(CommandBehavior) [page 71]	
public override unsafe int	ExecuteNonQuery() [page 71]	Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.
public new ULDataReader	ExecuteReader [page 72]	Executes a SQL SELECT statement and returns the result set.
public ULResultSet	ExecuteResultSet [page 76]	UL Ext: Executes a SQL SELECT statement and returns the result set as a ULResultSet object.
public override object	ExecuteScalar() [page 79]	Executes a SQL SELECT statement and returns a single value.

Modifier and Type	Method	Description
public ULTable	ExecuteTable [page 81]	UL Ext: Retrieves a database table in a ULTable object for direct manipulation.
public override unsafe void	Prepare() [page 84]	Pre-compiles and stores the SQL statement of this command.

Properties

Modifier and Type	Property	Description
public override string	CommandText [page 85]	Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType property is System.Data.CommandType.TableDirect.
public override int	CommandTimeout [page 86]	This feature is not supported by UltraLite.NET.
public override CommandType	CommandType [page 86]	Specifies the type of command to be executed.
public new ULConnection	Connection [page 87]	The connection object on which to execute the ULCommand object.
protected override DbConnection	DbConnection [page 88]	
protected override DbParameterCollection	DbParameterCollection [page 88]	
protected override DbTransaction	DbTransaction [page 88]	
public override bool	DesignTimeVisible [page 89]	Indicates if the ULCommand object should be visible in a customized Windows Form Designer control.
public string	IndexName [page 89]	UL Ext: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType property is System.Data.CommandType.TableDirect.
public new ULParameterCollection	Parameters [page 90]	Specifies the parameters for the current statement.
public unsafe string	Plan [page 91]	UL Ext: Returns the access plan UltraLite.NET uses to execute a query.
public new ULTransaction	Transaction [page 91]	Specifies the ULTransaction object in which the ULCommand object executes.
public override UpdateRowSource	UpdatedRowSource [page 92]	Specifies how command results are applied to the DataRow when used by the ULDataAdapterUpdate method.

Remarks

This object can be used to execute a statement or query efficiently multiple times.

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.

The ULCommand class 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 object. 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 object. 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 object for direct manipulation. The ULCommand.CommandText property is interpreted as the name of the table and the ULCommand.IndexName property can be used to specify a table sorting order. The ULCommand.CommandType property must be System.Data.CommandType.TableDirect.

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

For resource management reasons, you should 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.

In this section:

Provides a ULParameter object for supplying parameters to ULCommand objects.

Dispose(bool) Method [page 63]
Releases the unmanaged resources used by the ULCommand object and optionally releases the managed resources.

EndExecuteNonQuery(IAsyncResult) Method [page 64]
Finishes asynchronous execution of a SQL statement.

EndExecuteReader(IAsyncResult) Method [page 68]
Finishes asynchronous execution of a SQL statement, returning the requested ULDataReader.

ExecuteDbDataReader(CommandBehavior) Method [page 71]

ExecuteNonQuery() Method [page 71]
Executes a statement that does not return a result set, such as a SQL INSERT, DELETE, or UPDATE statement.

ExecuteReader Method [page 72]
Executes a SQL SELECT statement and returns the result set.

ExecuteResultSet Method [page 76]
UL Ext: Executes a SQL SELECT statement and returns the result set as a ULResultSet object.

ExecuteScalar() Method [page 79]
Executes a SQL SELECT statement and returns a single value.

ExecuteTable Method [page 81]
UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

Prepare() Method [page 84]
Pre-compiles and stores the SQL statement of this command.

CommandText Property [page 85]
Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType property is System.Data.CommandType.TableDirect.

CommandTimeout Property [page 86]
This feature is not supported by UltraLite.NET.

CommandType Property [page 86]
Specifies the type of command to be executed.

Connection Property [page 87]
The connection object on which to execute the ULCommand object.

DbConnection Property [page 88]

DbParameterCollection Property [page 88]

DbTransaction Property [page 88]

DesignTimeVisible Property [page 89]
Indicates if the ULCommand object should be visible in a customized Windows Form Designer control.

IndexName Property [page 89]
UL Ext: Specifies the name of the index to open (sort) the table with when the ULCommand.CommandType property is System.Data.CommandType.TableDirect.

Parameters Property [page 90]
Specifies the parameters for the current statement.

Plan Property [page 91]
UL Ext: Returns the access plan UltraLite.NET uses to execute a query.

Transaction Property [page 91]
Specifies the ULTransaction object in which the ULCommand object executes.

UpdatedRowSource Property [page 92]
Specifies how command results are applied to the DataRow when used by the ULDataAdapterUpdate method.

Related Information

CreateCommand() Method [page 129]
Transaction Property [page 91]
ExecuteNonQuery() Method [page 71]
ExecuteReader() Method [page 73]
ExecuteResultSet() Method [page 76]
ExecuteScalar() Method [page 79]
ExecuteTable() Method [page 81]
ULTable Class [page 507]
CommandText Property [page 85]
CommandType Property [page 86]
IndexName Property [page 89]
ULResultSet Class [page 421]
ULDataReader Class [page 260]

1.1.5.1 ULCommand Constructor

Initializes a ULCommand object.

Overload List

Modifier and Type	Overload name	Description
public	ULCommand() [page 51]	Initializes a ULCommand object.
public	ULCommand(string) [page 52]	Initializes a ULCommand object with the specified command text.
public	ULCommand(string, ULConnection) [page 53]	Initializes a ULCommand object with the specified command text and connection.

Modifier and Type	Overload name	Description
public	ULCommand(string, ULConnection, ULTransaction) [page 54]	Initializes a ULCommand object with the specified command text, connection, and transaction.

In this section:

ULCommand() Constructor [page 51]
Initializes a ULCommand object.

ULCommand(string) Constructor [page 52]
Initializes a ULCommand object with the specified command text.

ULCommand(string, ULConnection) Constructor [page 53]
Initializes a ULCommand object with the specified command text and connection.

ULCommand(string, ULConnection, ULTransaction) Constructor [page 54]
Initializes a ULCommand object with the specified command text, connection, and transaction.

1.1.5.1.1 ULCommand() Constructor

Initializes a ULCommand object.

Syntax

Visual Basic

```
Public Sub ULCommand ()
```

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.

Related Information

CreateCommand() Method [page 129]
ULCommand Class [page 45]
ULCommand Class [page 45]
ULCommand Class [page 45]

CommandText Property [page 85]
Connection Property [page 87]
Transaction Property [page 91]

1.1.5.1.2 ULCommand(string) Constructor

Initializes a ULCommand object with the specified command text.

Syntax

Visual Basic

```
Public Sub ULCommand (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 property 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 properties set before a statement can be executed.

Related Information

CreateCommand() Method [page 129]
ULCommand() Constructor [page 51]
ULCommand Class [page 45]
ULCommand Class [page 45]
Connection Property [page 87]
Transaction Property [page 91]
CommandType Property [page 86]

1.1.5.1.3 ULCommand(string, ULConnection) Constructor

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

Syntax

Visual Basic

```
Public Sub ULCommand (
    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 property 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 property set before a statement can be executed.

Related Information

ULConnection Class [page 108]
CreateCommand() Method [page 129]
ULCommand() Constructor [page 51]
ULCommand Class [page 45]
ULCommand Class [page 45]
Transaction Property [page 91]
CommandType Property [page 86]

1.1.5.1.4 ULCommand(string, ULConnection, ULTransaction) Constructor

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

Syntax

Visual Basic

```
Public Sub ULCommand (
    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 property 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 object in which the ULCommand object executes.

Related Information

CreateCommand() Method [page 129]
ULCommand() Constructor [page 51]
ULCommand Class [page 45]
ULCommand Class [page 45]
CommandType Property [page 86]
ULTransaction Class [page 557]

1.1.5.2 BeginExecuteNonQuery Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object.

Overload List

Modifier and Type	Overload name	Description
public IAsyncResult	BeginExecuteNonQuery() [page 55]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object.
public IAsyncResult	BeginExecuteNonQuery(AsyncCallback, object) [page 56]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, given a callback procedure and state information.

In this section:

BeginExecuteNonQuery() Method [page 55]
Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object.

BeginExecuteNonQuery(AsyncCallback, object) Method [page 56]
Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, given a callback procedure and state information.

1.1.5.2.1 BeginExecuteNonQuery() Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object.

Syntax

Visual Basic

```
Public Function BeginExecuteNonQuery () As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteNonQuery ()
```

Returns

A System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteNonQuery(IAsyncResult) method, which returns the number of affected rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

EndExecuteNonQuery(IAsyncResult) Method [page 64]
EndExecuteNonQuery(IAsyncResult) Method [page 64]

1.1.5.2.2 BeginExecuteNonQuery(AsyncCallback, object) Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, 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 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.

Returns

A System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteNonQuery(IAsyncResult) method, which returns the number of affected rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

EndExecuteNonQuery(IAsyncResult) Method [page 64]

1.1.5.3 BeginExecuteReader Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set.

Overload List

Modifier and Type	Overload name	Description
public IAsyncResult	BeginExecuteReader() [page 58]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set.
public IAsyncResult	BeginExecuteReader(AsyncCallback, object) [page 59]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set, given a callback procedure and state information.
public IAsyncResult	BeginExecuteReader(AsyncCallback, object, CommandBehavior) [page 60]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, using one of the CommandBehavior values, and retrieves the result set, given a callback procedure and state information.
public IAsyncResult	BeginExecuteReader(CommandBehavior) [page 61]	Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, using one of the CommandBehavior values, and retrieves the result set.

In this section:

BeginExecuteReader() Method [page 58]
Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set.

BeginExecuteReader(AsyncCallback, object) Method [page 59]
: Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set, given a callback procedure and state information.

BeginExecuteReader(AsyncCallback, object, CommandBehavior) Method [page 60]
: Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, using one of the CommandBehavior values, and retrieves the result set, given a callback procedure and state information.

BeginExecuteReader(CommandBehavior) Method [page 61]
: Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, using one of the CommandBehavior values, and retrieves the result set.

1.1.5.3.1 BeginExecuteReader() Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, and retrieves the result set.

Syntax

Visual Basic

```
Public Function BeginExecuteReader () As IAsyncResult
```

C#

```
public IAsyncResult BeginExecuteReader ()
```

Returns

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteReader(IAsyncResult) method, which returns an ULDataReader object that can be used to retrieve the returned rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

EndExecuteReader(IAsyncResult) Method [page 68]
ULDataReader Class [page 260]

1.1.5.3.2 BeginExecuteReader(AsyncCallback, object) Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, 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.

Returns

A System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteReader(IAsyncResult) method, which returns a ULDataReader object that can be used to retrieve the returned rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

ULDataReader Class [page 260]

EndExecuteReader(IAsyncResult) Method [page 68]
EndExecuteReader(IAsyncResult) Method [page 68]

1.1.5.3.3 BeginExecuteReader(AsyncCallback, object, CommandBehavior) Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, 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.

Returns

A System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteReader(IAsyncResult) method, which returns a ULDataReader object that can be used to retrieve the returned rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

EndExecuteReader(IAsyncResult) Method [page 68]
ULDataReader Class [page 260]

1.1.5.3.4 BeginExecuteReader(CommandBehavior) Method

Initiates the asynchronous execution of a SQL statement that is described by this ULCommand object, 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.

Returns

An System.IAsyncResult that can be used to poll, wait for results, or both is returned; this value is also needed when invoking the EndExecuteReader(IAsyncResult) method, which returns a ULDataReader object that can be used to retrieve the returned rows.

Exceptions

ULException class Any error that occurred while executing the command text.

Related Information

EndExecuteReader(IAsyncResult) Method [page 68]
ULDataReader Class [page 260]

1.1.5.4 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.

1.1.5.5 CreateDbParameter() Method

Syntax

Visual Basic

```
Protected Overrides Function CreateDbParameter () As DbParameter
```

C#

```
protected override DbParameter CreateDbParameter ()
```

1.1.5.6 CreateParameter() Method

Provides a ULParameter object for supplying parameters to ULCommand objects.

Syntax

Visual Basic

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

C#

```
public new ULParameter CreateParameter ()
```

Returns

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 object to specify the value for the parameter.

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

Related Information

ULParameter Class [page 372]

1.1.5.7 Dispose(bool) Method

Releases the unmanaged resources used by the ULCommand object and optionally releases the managed resources.

Syntax

Visual Basic

```
Protected Overrides Sub Dispose (ByVal disposing As Boolean)
```

C#

```
protected override void Dispose (bool disposing)
```

Parameters

disposing When true, disposes of both the managed and unmanaged resources. When false, disposes of only the unmanaged resources.

1.1.5.8 EndExecuteNonQuery(IAsyncResult) 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 the BeginExecuteNonQuery method.

Returns

The number of rows affected, which is the same behavior as the ExecuteNonQuery method.

Exceptions

ArgumentException The asyncResult parameter is null (Nothing in Microsoft Visual Basic).
InvalidOperationException The EndExecuteNonQuery(IAsyncResult) method was called more than once for a single command execution, or the method was mismatched against its execution method.

Remarks

You must call the EndExecuteNonQuery method once for every BeginExecuteNonQuery call. The call must be made after the BeginExecuteNonQuery call returns. ADO.NET is not thread safe; you must ensure that the BeginExecuteNonQuery call has returned. The System.IAsyncResult passed to the EndExecuteNonQuery method must be the same as the one returned from the BeginExecuteNonQuery call that is being completed. It is an error to call the EndExecuteNonQuery method to end a call to the BeginExecuteReader method, and vice versa.

If an error occurs while executing the command, the exception is thrown when the EndExecuteNonQuery method 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 blocks until the command completes.
Dim rowCount As Integer = _
   cmd.EndExecuteNonQuery( res )
```

The following code is the C# language equivalent:

```
// C#
ULCommand cmd = new ULCommand(
   "UPDATE Departments"
   + " SET DepartmentName = 'Engineering'"
   + " WHERE DepartmentID=100",
   conn
);
IAsyncResult res = cmd.BeginExecuteNonQuery();
// Perform other work.
// This blocks 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 )
   ' Perform other work.
End While
' This blocks until the command completes.
Dim rowCount As Integer = _
```

```
    cmd.EndExecuteNonQuery( res )
```

The following code is the C# language equivalent:

```
// C#
ULCommand cmd = new ULCommand(
   "UPDATE Departments"
   + " SET DepartmentName = 'Engineering'"
   + " WHERE DepartmentID=100",
   conn
);
IAsyncResult res = cmd.BeginExecuteNonQuery();
while( !res.IsCompleted ) {
   // Perform other work.
}
// This blocks 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 does not block because the command is finished.
Dim rowCount As Integer = _
 cmd.EndExecuteNonQuery( res )
```

The following code is the C# language equivalent:

```
// 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 does not block because the command is finished.
int rowCount = cmd.EndExecuteNonQuery( res );
```

Specify a callback function when calling the BeginExecuteNonQuery method You can specify a callback function when calling the BeginExecuteNonQuery method. 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
   ' is called when the command completes.
End Sub
```

The following code is the C# language equivalent:

```
// 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
   // is 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.

Related Information

BeginExecuteNonQuery() Method [page 55]

1.1.5.9 EndExecuteReader(IAsyncResult) 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 BeginExecuteReader call.

Returns

An ULDataReader object that can be used to retrieve the requested rows, which is the same behavior as the ExecuteReader method.

Exceptions

ArgumentException The asyncResult parameter is null (Nothing in Microsoft Visual Basic)

InvalidOperationException The EndExecuteReader method was called more than once for a single command execution, or the method was mismatched against its execution method.

Remarks

You must call the EndExecuteReader method once for every call to the BeginExecuteReader method. The call must be after the BeginExecuteReader call returns. ADO.NET is not thread safe; it is your responsibility to ensure that the BeginExecuteReader method has returned. The System.IAsyncResult passed to the EndExecuteReader method must be the same as the one returned from the BeginExecuteReader call that is being completed. It is an error to call the EndExecuteReader method to end a BeginExecuteNonQuery call, and vice versa.

If an error occurs while executing the command, the exception is thrown when the EndExecuteReader method is called.

There are four ways to wait for execution to complete:

Call the EndExecuteReader method Calling the EndExecuteReader method 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 blocks until the command completes.
Dim reader As ULDataReader = _
   cmd.EndExecuteReader( res )
```

The following code is the C# language equivalent:

```
// C#
ULCommand cmd = new ULCommand(
   "SELECT * FROM Departments", conn
);
IAsyncResult res = cmd.BeginExecuteReader();
// Perform other work
// This blocks 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 )
   ' Perform other work
End While
' This blocks 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 ) {
   // Perform other work.
}
// This blocks 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 does not block because the command is finished.
Dim reader As ULDataReader = _
   cmd.EndExecuteReader( res )
```

The following code is the C# language equivalent:

```
// C#
ULCommand cmd = new ULCommand(
   "SELECT * FROM Departments", conn
);
IAsyncResult res = cmd.BeginExecuteReader();
// Perform other work.
WaitHandle wh = res.AsyncWaitHandle;
wh.WaitOne();
// This does not block because the command is finished.
ULDataReader reader = cmd.EndExecuteReader( res );
```

Specify a callback function when calling the BeginExecuteReader method You can specify a callback function when calling the BeginExecuteReader method. 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
   ' is called when the command completes.
End Sub
```

The following code is the C# language equivalent:

```
// 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
   // is 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.

Related Information

BeginExecuteReader() Method [page 58]
ULDataReader Class [page 260]

1.1.5.10 ExecuteDbDataReader(CommandBehavior) Method

Syntax

Visual Basic

```
Protected Overrides Function ExecuteDbDataReader (ByVal cmdBehavior As
CommandBehavior) As DbDataReader
```

C#

```
protected override DbDataReader ExecuteDbDataReader (CommandBehavior
cmdBehavior)
```

1.1.5.11 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 unsafe int ExecuteNonQuery ()
```

Returns

The number of rows affected.

Exceptions

ULException class A SQL error occurred.

InvalidOperationException The command is in an invalid state. Either the ULCommand.Connection object is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText value is invalid.

Remarks

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

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 property cannot be System.Data.CommandType.TableDirect.

Related Information

CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
Parameters Property [page 90]
Transaction Property [page 91]

1.1.5.12 ExecuteReader Method

Executes a SQL SELECT statement and returns the result set.

Overload List

Modifier and Type	Overload name	Description
public new ULDataReader	ExecuteReader() [page 73]	Executes a SQL SELECT statement and returns the result set.
public new ULDataReader	ExecuteReader(CommandBehavior) [page 74]	Executes a SQL SELECT statement with the specified command behavior and returns the result set.

In this section:

ExecuteReader() Method [page 73]
Executes a SQL SELECT statement and returns the result set.

ExecuteReader(CommandBehavior) Method [page 74]
Executes a SQL SELECT statement with the specified command behavior and returns the result set.

1.1.5.12.1 ExecuteReader() Method

Executes a SQL SELECT statement and returns the result set.

Syntax

Visual Basic

```
Public Shadows Function ExecuteReader () As ULDataReader
```

C#

```
public new ULDataReader ExecuteReader ()
```

Returns

The result set as a ULDataReader object.

Exceptions

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

Remarks

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

If the ULCommand.CommandType value is System.Data.CommandType.TableDirect, the ExecuteReader method performs a ULCommand.ExecuteTable call and returns a ULTable object downcast as a ULDataReader object.

SELECT statements are marked as read-only by default for performance reasons. If the query is going to be used to make updates, the statement must end with " FOR UPDATE".

This is the strongly-typed version of the System.Data.IDbCommand.ExecuteReader and System.Data.Common.DbCommand.ExecuteReader methods.

Related Information

CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
Parameters Property [page 90]
ExecuteResultSet() Method [page 76]
ExecuteTable() Method [page 81]
Transaction Property [page 91]
ULDataAdapter Class [page 224]
ULDataReader Class [page 260]
ULTable Class [page 507]

1.1.5.12.2 ExecuteReader(CommandBehavior) Method

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

Syntax

Visual Basic

```
Public Shadows Function ExecuteReader (ByVal cmdBehavior As
CommandBehavior) As ULDataReader
```

C#

```
public new 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.

Returns

The result set as a ULDataReader object.

Exceptions

ULException class A SQL error occurred.

InvalidOperationException The command is in an invalid state. Either the ULCommand.Connection value is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText value is invalid.

Remarks

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

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

SELECT statements are marked as read-only by default for performance reasons. If the query is going to be used to make updates, the statement must end with " FOR UPDATE".

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

Related Information

CommandText Property [page 85]
Connection Property [page 87]
ExecuteReader() Method [page 73]
ExecuteResultSet(CommandBehavior) Method [page 78]
ExecuteTable(CommandBehavior) Method [page 82]
Parameters Property [page 90]
Transaction Property [page 91]
ULDataAdapter Class [page 224]
ULDataReader Class [page 260]
ULTable Class [page 507]
ULDataReader Class [page 260]

1.1.5.13 ExecuteResultSet Method

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

Overload List

Modifier and Type	Overload name	Description
public ULResultSet	ExecuteResultSet() [page 76]	UL Ext: Executes a SQL SELECT statement and returns the result set as a ULResultSet object.
public ULResultSet	ExecuteResultSet(CommandBehavior) [page 78]	UL Ext: Executes a SQL SELECT statement with the specified command behavior and returns the result set as a ULResultSet object.

In this section:

ExecuteResultSet() Method [page 76]
UL Ext: Executes a SQL SELECT statement and returns the result set as a ULResultSet object.

ExecuteResultSet(CommandBehavior) Method [page 78]
UL Ext: Executes a SQL SELECT statement with the specified command behavior and returns the result set as a ULResultSet object.

1.1.5.13.1 ExecuteResultSet() Method

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

Syntax

Visual Basic

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

C#

```
public ULResultSet ExecuteResultSet ()
```

Returns

The result set as a ULResultSet object.

Exceptions

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

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText and any ULCommand.Parameters values 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 the ULCommand.ExecuteTable method or a ULDataAdapter object.

If the ULCommand.CommandType value is System.Data.CommandType.TableDirect, the ExecuteReader method performs a ULCommand.ExecuteTable call and returns a ULTable object downcast as a ULResultSet object.

This method supports positioned updates and deletes with Dynamic SQL.

Example

```
cmd.CommandText = "SELECT id, season, price FROM OurProducts";
ULResultSet rs = cmd.ExecuteResultSet();
while( rs.Read() ) {
   string season = rs.GetString( 1 );
   double price = rs.GetDouble( 2 );
   if( season.Equals( "summer" ) ) {
       rs.UpdateBegin();
       rs.SetDouble( 2, price * .5 );
       rs.Update();
   }
   if( season.Equals( "discontinued" ) ) {
       rs.Delete();
   }
}
rs.Close();
```

Related Information

ULCommand Class [page 45]
CommandText Property [page 85]
Parameters Property [page 90]
CommandType Property [page 86]
ULResultSet Class [page 421]

ULTable Class [page 507]

1.1.5.13.2 ExecuteResultSet(CommandBehavior) Method

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

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.

Returns

The result set as a ULResultSet object.

Exceptions

ULException class A SQL error occurred.

InvalidOperationException The command is in an invalid state. Either the ULCommand.Connection value is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText value is invalid.

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText value and any ULCommand.Parameters value 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 the ULCommand.ExecuteTable(CommandBehavior) method or a ULDataAdapter object.

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

This method supports positioned updates and deletes with Dynamic SQL.

Related Information

ULResultSet Class [page 421]
CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
ExecuteReader() Method [page 73]
ExecuteTable(CommandBehavior) Method [page 82]
Parameters Property [page 90]
Transaction Property [page 91]
ULDataAdapter Class [page 224]
ULTable Class [page 507]

1.1.5.14 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 ()
```

Returns

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.

Exceptions

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

Remarks

The statement is the current ULCommand object, with the ULCommand.CommandText value and any ULCommand.Parameters value 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 value is System.Data.CommandType.TableDirect, the ExecuteScalar method performs a ULCommand.ExecuteTable call and returns the first column of the first row.

SELECT statements are marked as read-only by default for performance reasons. If the query is going to be used to make updates, the statement must end with " FOR UPDATE".

Related Information

CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
IndexName Property [page 89]
Parameters Property [page 90]
Transaction Property [page 91]

1.1.5.15 ExecuteTable Method

UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

Overload List

Modifier and Type	Overload name	Description
public ULTable	ExecuteTable() [page 81]	UL Ext: Retrieves a database table in a ULTable object for direct manipulation.
public ULTable	ExecuteTable(CommandBehavior) [page 82]	UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

In this section:

ExecuteTable() Method [page 81]
UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

ExecuteTable(CommandBehavior) Method [page 82]
UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

1.1.5.15.1 ExecuteTable() Method

UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

Syntax

Visual Basic

```
Public Function ExecuteTable () As ULTable
```

C#

```
public ULTable ExecuteTable ()
```

Returns

The table as a ULTable object.

Exceptions

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

Remarks

The ULCommand.CommandText value is interpreted as the name of the table, and ULCommand.IndexName value can be used to specify a table sorting order.

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

If the ULCommand.IndexName value 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.

Related Information

CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
IndexName Property [page 89]
Transaction Property [page 91]
ULTable Class [page 507]

1.1.5.15.2 ExecuteTable(CommandBehavior) Method

UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

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.

Returns

The table as a ULTable object.

Exceptions

ULException class A SQL error occurred.

InvalidOperationException The command is in an invalid state. Either the ULCommand.Connection value is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText value is invalid.

Remarks

The ULCommand.CommandText value is interpreted as the name of the table, and ULCommand.IndexName value can be used to specify a table sorting order.

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

If the ULCommand.IndexName value 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.

Related Information

CommandText Property [page 85]
CommandType Property [page 86]
Connection Property [page 87]
ExecuteTable() Method [page 81]
IndexName Property [page 89]
Transaction Property [page 91]

1.1.5.16 Prepare() Method

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

Syntax

Visual Basic

```
Public Overrides Sub Prepare ()
```

C#

```
public override unsafe void Prepare ()
```

Exceptions

ULException class A SQL error occurred.

InvalidOperationException The command is in an invalid state. Either the ULCommand.Connection value is missing or closed, the ULCommand.Transaction value does not match the current transaction state of the connection, or the ULCommand.CommandText value is invalid.

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.

Related Information

Connection Property [page 87]
Transaction Property [page 91]
CommandText Property [page 85]

1.1.5.17 CommandText Property

Specifies the text of the SQL statement or the name of the table when the ULCommand.CommandType property is System.Data.CommandType.TableDirect.

Syntax

Visual Basic

```
Public Overrides Property CommandText As String
```

C#

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

Remarks

For parameterized statements, use a question mark (?) placeholder to pass parameters.

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

SELECT statements are marked as read-only by default for performance reasons. If the query is going to be used to make updates, the statement must end with " FOR UPDATE".

Example

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

The following code is the C# language equivalent:

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

Related Information

ExecuteNonQuery() Method [page 71]
ExecuteReader() Method [page 73]
ExecuteResultSet() Method [page 76]
ExecuteScalar() Method [page 79]
ExecuteTable() Method [page 81]
CommandType Property [page 86]

1.1.5.18 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;}
```

Remarks

The value is always zero.

1.1.5.19 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;}
```

Remarks

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

Supported command types are as follows:

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

ULCommand.ExecuteNonQuery method to execute a non-query SQL statement, and use either the ULCommand.ExecuteReader or ULCommand.ExecuteScalar method to execute a query.

Related Information

CommandText Property [page 85]
IndexName Property [page 89]
ExecuteTable() Method [page 81]
ExecuteReader() Method [page 73]
CommandText Property [page 85]
ExecuteNonQuery() Method [page 71]
ExecuteReader() Method [page 73]
ExecuteScalar() Method [page 79]

1.1.5.20 Connection Property

The connection object on which to execute the ULCommand object.

Syntax

Visual Basic

```
Public Shadows Property Connection As ULConnection
```

C#

```
public new ULConnection Connection {get;set;}
```

Remarks

The ULConnection object on which to execute the command.

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.

Related Information

ULConnection Class [page 108]

1.1.5.21 DbConnection Property

Syntax

Visual Basic

```
Protected Overrides Property DbConnection As DbConnection
```

C#

```
protected override DbConnection DbConnection {get;set;}
```

1.1.5.22 DbParameterCollection Property

Syntax

Visual Basic

```
Protected ReadOnly Overrides Property DbParameterCollection As
DbParameterCollection
```

C#

```
protected override DbParameterCollection DbParameterCollection {get;}
```

1.1.5.23 DbTransaction Property

Syntax

Visual Basic

```
Protected Overrides Property DbTransaction As DbTransaction
```

C#

```
protected override DbTransaction DbTransaction {get;set;}
```

1.1.5.24 DesignTimeVisible Property

Indicates if the ULCommand object 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;}
```

Remarks

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

1.1.5.25 IndexName Property

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

Syntax

Visual Basic

```
Public Property IndexName As String
```

C#

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

Remarks

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.

Related Information

ExecuteTable() Method [page 81]

ExecuteReader() Method [page 73]
CommandType Property [page 86]

1.1.5.26 Parameters Property

Specifies the parameters for the current statement.

Syntax

Visual Basic

```
Public ReadOnly Shadows Property Parameters As ULParameterCollection
```

C#

```
public new ULParameterCollection Parameters {get;}
```

Remarks

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

Use question marks in ULCommand.CommandText property value 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 property as there are parameters in this collection.

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

Related Information

ULParameterCollection Class [page 392]
ULParameter Class [page 372]
CommandText Property [page 85]

1.1.5.27 Plan Property

UL Ext: Returns the access plan UltraLite.NET uses to execute a query.

Syntax

Visual Basic

```
Public ReadOnly Property Plan As String
```

C#

```
public unsafe string Plan {get;}
```

Remarks

This property is intended primarily for use during development.

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

1.1.5.28 Transaction Property

Specifies the ULTransaction object in which the ULCommand object executes.

Syntax

Visual Basic

```
Public Shadows Property Transaction As ULTransaction
```

C#

```
public new ULTransaction Transaction {get;set;}
```

Remarks

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

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.

Related Information

BeginTransaction() Method [page 121]
ULTransaction Class [page 557]
Connection Property [page 87]

1.1.5.29 UpdatedRowSource Property

Specifies how command results are applied to the DataRow when used by the ULDataAdapterUpdate method.

Syntax

Visual Basic

```
Public Overrides Property UpdatedRowSource As UpdateRowSource
```

C#

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

Remarks

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

1.1.6 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 System.Data.Common.DbCommandBuilder
```

C#

```
public class ULCommandBuilder : System.Data.Common.DbCommandBuilder
```

Members

All members of ULCommandBuilder, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULCommandBuilder [page 95]	Initializes a ULCommandBuilder object.

Methods

Modifier and Type	Method	Description
protected override void	ApplyParameterInfo(DbParameter, DataRow, StatementType, bool) [page 97]	
public new ULCommand	GetDeleteCommand [page 97]	Gets the automatically generated ULCommand object required to perform deletions on the database.
public new ULCommand	GetInsertCommand [page 100]	Gets the automatically generated ULCommand object required to perform insertions on the database.
protected override string	GetParameterName [page 103]	See individual topics
protected override string	GetParameterPlaceholder(int) [page 104]	
public new ULCommand	GetUpdateCommand [page 104]	Gets the automatically generated ULCommand object required to perform updates on the database.
protected override DbCommand	InitializeCommand(DbCommand) [page 107]	
protected override void	SetRowUpdatingHandler(DbDataAdapter) [page 107]	

Properties

Modifier and Type	Property	Description
public new ULDataAdapter	DataAdapter [page 108]	Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.

Remarks

The ULDataAdapter object 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 object. Then, any additional SQL statements that you do not set are generated by the ULCommandBuilder object.

Example

The following example uses the ULCommand object, along with the ULDataAdapter and ULConnection objects, 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 object.

```
' 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)
   'Insert code to modify data in DataSet.
   'Without the ULCommandBuilder this line would fail
   adapter.Update(dataSet, tableName)
   Return dataSet
End Function
```

The following code is the C# language equivalent:

```
// 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);
       // Insert code to modify data in DataSet.
       // Without the ULCommandBuilder this line would fail
       adapter.Update(dataSet, tableName);
       return dataSet;
   }
}
```

In this section:

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

InitializeCommand(DbCommand) Method [page 107]

SetRowUpdatingHandler(DbDataAdapter) Method [page 107]

DataAdapter Property [page 108]
Gets or sets a ULDataAdapter object for which SQL statements are automatically generated.

Related Information

ULCommand Class [page 45]
ULConnection Class [page 108]
ULDataAdapter Class [page 224]

1.1.6.1 ULCommandBuilder Constructor

Initializes a ULCommandBuilder object.

Overload List

Modifier and Type	Overload name	Description
public	ULCommandBuilder() [page 96]	Initializes a ULCommandBuilder object.
public	ULCommandBuilder(ULDataAdapter) [page 96]	Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

In this section:

ULCommandBuilder() Constructor [page 96]
Initializes a ULCommandBuilder object.

ULCommandBuilder(ULDataAdapter) Constructor [page 96]
Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

1.1.6.1.1 ULCommandBuilder() Constructor

Initializes a ULCommandBuilder object.

Syntax

Visual Basic

```
Public Sub ULCommandBuilder ()
```

C#

```
public ULCommandBuilder ()
```

Related Information

ULCommandBuilder(ULDataAdapter) Constructor [page 96]

1.1.6.1.2 ULCommandBuilder(ULDataAdapter) Constructor

Initializes a ULCommandBuilder object with the specified ULDataAdapter object.

Syntax

Visual Basic

```
Public Sub ULCommandBuilder (ByVal adapter As ULDataAdapter)
```

C#

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

Parameters

adapter A ULDataAdapter object.

Related Information

ULDataAdapter Class [page 224]

1.1.6.2 ApplyParameterInfo(DbParameter, DataRow, StatementType, bool) Method

Syntax

Visual Basic

```
Protected Overrides Sub ApplyParameterInfo (
    ByVal parameter As DbParameter,
    ByVal row As DataRow,
    ByVal statementType As StatementType,
    ByVal whereClause As Boolean
)
```

C#

```
protected override void ApplyParameterInfo (
    DbParameter parameter,
    DataRow row,
    StatementType statementType,
    bool whereClause
)
```

1.1.6.3 GetDeleteCommand Method

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

Overload List

Modifier and Type	Overload name	Description
public new ULCommand	GetDeleteCommand() [page 98]	Gets the automatically generated ULCommand object required to perform deletions on the database.
public new ULCommand	GetDeleteCommand(bool) [page 99]	Gets the automatically generated ULCommand object required to perform deletions on the database.

In this section:

GetDeleteCommand() Method [page 98]
Gets the automatically generated ULCommand object required to perform deletions on the database.

GetDeleteCommand(bool) Method [page 99]
Gets the automatically generated ULCommand object required to perform deletions on the database.

1.1.6.3.1 GetDeleteCommand() Method

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

Syntax

Visual Basic

```
Public Shadows Function GetDeleteCommand () As ULCommand
```

C#

```
public new ULCommand GetDeleteCommand ()
```

Returns

The automatically generated ULCommand object required to perform deletions.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 value that contains duplicate columns.Dynamic SQL generation for the DeleteCommand property is not supported against a SelectCommand value that does not return any key column information.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema method if it changes the ULDataAdapter.SelectCommand value in any way. Otherwise, the GetDeleteCommand method still uses 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 GetDeleteCommand methods.

Related Information

ULCommand Class [page 45]
SelectCommand Property [page 236]

1.1.6.3.2 GetDeleteCommand(bool) Method

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

Syntax

Visual Basic

```
Public Shadows Function GetDeleteCommand (ByVal
useColumnsForParameterNames As Boolean) As ULCommand
```

C#

```
public new ULCommand GetDeleteCommand (bool useColumnsForParameterNames)
```

Parameters

useColumnsForParameterNames If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Returns

The automatically generated ULCommand object required to perform deletions.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 value that contains duplicate columns.Dynamic SQL generation for the DeleteCommand property is not supported against a SelectCommand value that does not return any key column information.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema method if it changes the ULDataAdapter.SelectCommand value in any way. Otherwise, the GetDeleteCommand method still uses 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 GetDeleteCommand methods.

Related Information

ULCommand Class [page 45]
SelectCommand Property [page 236]

1.1.6.4 GetInsertCommand Method

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

Overload List

Modifier and Type	Overload name	Description
public new ULCommand	GetInsertCommand() [page 100]	Gets the automatically generated ULCommand object required to perform insertions on the database.
public new ULCommand	GetInsertCommand(bool) [page 101]	Gets the automatically generated ULCommand object required to perform insertions on the database.

In this section:

GetInsertCommand() Method [page 100]
Gets the automatically generated ULCommand object required to perform insertions on the database.

GetInsertCommand(bool) Method [page 101]
Gets the automatically generated ULCommand object required to perform insertions on the database.

1.1.6.4.1 GetInsertCommand() Method

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

Syntax

Visual Basic

```
Public Shadows Function GetInsertCommand () As ULCommand
```

C#

```
public new ULCommand GetInsertCommand ()
```

Returns

The automatically generated ULCommand object required to perform insertions.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 property is not supported against a SelectCommand value 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 value that contains duplicate columns.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema method if it changes the ULDataAdapter.SelectCommand value in any way. Otherwise, the GetInsertCommand method still uses 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 GetInsertCommand methods.

Related Information

ULCommand Class [page 45]
SelectCommand Property [page 236]

1.1.6.4.2 GetInsertCommand(bool) Method

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

Syntax

Visual Basic

```
Public Shadows Function GetInsertCommand (ByVal
useColumnsForParameterNames As Boolean) As ULCommand
```

C#

```
public new ULCommand GetInsertCommand (bool useColumnsForParameterNames)
```

Parameters

useColumnsForParameterNames If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Returns

The automatically generated ULCommand object required to perform insertions.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 property is not supported against a SelectCommand value 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 value that contains duplicate columns.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema method if it changes the ULDataAdapter.SelectCommand value in any way. Otherwise, the GetInsertCommand method still uses 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 GetInsertCommand methods.

Related Information

ULCommand Class [page 45]
SelectCommand Property [page 236]

1.1.6.5 GetParameterName Method

Overload List

Modifier and Type	Overload name	Description
protected override string	GetParameterName(int) [page 103]	
protected override string	GetParameterName(string) [page 103]	

In this section:

1.1.6.5.1 GetParameterName(int) Method

Syntax

Visual Basic

```
Protected Overrides Function GetParameterName (ByVal index As Integer) As
String
```

C#

```
protected override string GetParameterName (int index)
```

1.1.6.5.2 GetParameterName(string) Method

Syntax

Visual Basic

```
Protected Overrides Function GetParameterName (ByVal parameterName As
String) As String
```

C#

```
protected override string GetParameterName (string parameterName)
```

1.1.6.6 GetParameterPlaceholder(int) Method

Syntax

Visual Basic

```
Protected Overrides Function GetParameterPlaceholder (ByVal index As
Integer) As String
```

C#

```
protected override string GetParameterPlaceholder (int index)
```

1.1.6.7 GetUpdateCommand Method

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

Overload List

Modifier and Type	Overload name	Description
public new ULCommand	GetUpdateCommand() [page 105]	Gets the automatically generated ULCommand object required to perform updates on the database.
public new ULCommand	GetUpdateCommand(bool) [page 106]	Gets the automatically generated ULCommand object required to perform updates on the database.

In this section:

GetUpdateCommand() Method [page 105]
Gets the automatically generated ULCommand object required to perform updates on the database.

GetUpdateCommand(bool) Method [page 106]
Gets the automatically generated ULCommand object required to perform updates on the database.

1.1.6.7.1 GetUpdateCommand() Method

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

Syntax

Visual Basic

```
Public Shadows Function GetUpdateCommand () As ULCommand
```

C#

```
public new ULCommand GetUpdateCommand ()
```

Returns

The automatically generated ULCommand object required to perform updates.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 property is not supported against a SelectCommand value 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 value that contains duplicate columns.Dynamic SQL generation for the UpdateCommand property is not supported against a SelectCommand value that does not return any key column information.

Remarks

After the SQL statement is first generated, the application must explicitly call the DbCommandBuilder.RefreshSchema method if it changes the ULDataAdapter.SelectCommand value in any way. Otherwise, the GetUpdateCommand method still uses 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 GetUpdateCommand methods.

Related Information

ULCommand Class [page 45]

SelectCommand Property [page 236]

1.1.6.7.2 GetUpdateCommand(bool) Method

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

Syntax

Visual Basic

```
Public Shadows Function GetUpdateCommand (ByVal
useColumnsForParameterNames As Boolean) As ULCommand
```

C#

```
public new ULCommand GetUpdateCommand (bool useColumnsForParameterNames)
```

Parameters

useColumnsForParameterNames If true, generate parameter names matching column names if possible. If false, generate @p1, @p2, and so on.

Returns

The automatically generated ULCommand object required to perform updates.

Exceptions

InvalidOperationException The DbCommandBuilder.DataAdapter property 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 property is not supported against a SelectCommand value 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 value that contains duplicate columns.Dynamic SQL generation for the UpdateCommand property is not supported against a SelectCommand value that does not return any key column information.

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 still uses 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 GetUpdateCommand methods.

Related Information

ULCommand Class [page 45]
SelectCommand Property [page 236]

1.1.6.8 InitializeCommand(DbCommand) Method

Syntax

Visual Basic

```
Protected Overrides Function InitializeCommand (ByVal command As
DbCommand) As DbCommand
```

C#

```
protected override DbCommand InitializeCommand (DbCommand command)
```

1.1.6.9 SetRowUpdatingHandler(DbDataAdapter) Method

Syntax

Visual Basic

```
Protected Overrides Sub SetRowUpdatingHandler (ByVal adapter As
DbDataAdapter)
```

C#

```
protected override void SetRowUpdatingHandler (DbDataAdapter adapter)
```

1.1.6.10 DataAdapter Property

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

Syntax

Visual Basic

```
Public Shadows Property DataAdapter As ULDataAdapter
```

C#

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

Remarks

A ULDataAdapter object.

Related Information

ULDataAdapter Class [page 224]

1.1.7 ULConnection Class

Represents a connection to an UltraLite.NET database.

Syntax

Visual Basic

```
Public NotInheritable Class ULConnection Inherits
System.Data.Common.DbConnection
```

C#

```
public sealed class ULConnection : System.Data.Common.DbConnection
```

Members

All members of ULConnection, including inherited members.

Variables

Modifier and Type	Variable	Description
public const long	INVALID_DATABASE_ID	UL Ext: A database ID constant indicating that the ULConnection.DatabaseID property has not been set.
public const String	SYNC_ALL_DB	Empty publication list, corresponding to the entire database.
public const String	SYNC_ALL_PUBS	Publication name "*", corresponding to all publications.

Constructors

Modifier and Type	Constructor	Description
public	ULConnection [page 115]	Initializes a ULConnection object.

Methods

Modifier and Type	Method	Description
protected override DbTransaction	BeginDbTransaction(IsolationLevel) [page 117]	
public IAsyncResult	BeginSynchronize [page 118]	UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.
public new ULTransaction	BeginTransaction [page 121]	Returns a transaction object.
public unsafe int	CancelGetNotification(string) [page 124]	UL Ext: Cancels any pending get-notification calls on all queues matching the given name.
public void	CancelSynchronize(IAsyncResult) [page 125]	UL Ext: Causes a running synchronization to be canceled at the next opportunity.
public override void	ChangeDatabase(string) [page 125]	Changes the current database for an open connection.
public void	ChangeEncryptionKey(string) [page 126]	UL Ext: Changes the database's encryption key to the specified new key.
public static void	ChangePassword(string, string) [page 127]	Changes the password for the user indicated in the connection string to the supplied new password.
public override void	Close() [page 128]	Closes the database connection.
public unsafe long	CountUploadRows(string, long) [page 128]	UL Ext: Returns the number of rows that need to be uploaded when the next synchronization takes place.
public new ULCommand	CreateCommand() [page 129]	Creates and initializes a ULCommand object associated with this connection and its current transaction.
protected override DbCommand	CreateDbCommand() [page 130]	

Modifier and Type	Method	Description
public void	CreateNotificationQueue(string, string) [page 130]	UL Ext: Creates an event queue.
public void	DeclareEvent(string) [page 131]	UL Ext: Declares a named event.
public void	DestroyNotificationQueue(string) [page 132]	UL Ext: Destroys an event queue.
protected override unsafe void	Dispose(bool) [page 133]	Releases the unmanaged resources used by the ULCommand object and optionally releases the managed resources.
public void	EndSynchronize(IAsyncResult) [page 133]	UL Ext: Blocks until an asynchronously launched synchronization terminates.
public ULTable	ExecuteTable [page 134]	UL Ext: Retrieves a database table in a ULTable object for direct manipulation.
public unsafe DateTime	GetLastDownloadTime(string) [page 140]	UL Ext: Returns the time of the most recent download of the specified publication.
public unsafe Guid	GetNewUUID() [page 141]	UL Ext: Generates a new UUID (System.Guid).
public string	GetNotification(string, int) [page 142]	UL Ext: Blocks for a notification or timeout.
public string	GetNotificationParameter(string, string) [page 143]	UL Ext: Gets the value of a parameter for an event that was just read by the GetNotification method.
public override DataTable	GetSchema [page 145]	Returns the list of supported schema collections.
public void	GrantConnectTo(string, string) [page 148]	UL Ext: Grants access to an UltraLite database for a user ID with a specified password.
public override void	Open() [page 149]	Opens a connection to a database using the previously-specified connection string.
public void	RegisterForEvent(string, string, string, bool) [page 150]	UL Ext: Registers a queue to get events from an object.
public void	ResetLastDownloadTime(string) [page 151]	UL Ext: Resets the time of the most recent download.
public void	RevokeConnectFrom(string) [page 152]	UL Ext: Revokes access to an UltraLite database from the specified user ID.
public void	RollbackPartialDownload() [page 152]	UL Ext: Rolls back outstanding changes to the database from a partial download.
public unsafe int	SendNotification(string, string, string) [page 153]	UL Ext: Sends a notification to matching queues.
public void	SetSyncListener(ULSyncProgressListener) [page 154]	Specifies the listener object used to process synchronization messages.

Modifier and Type	Method	Description
public void	StartSynchronizationDelete() [page 155]	UL Ext: Marks all subsequent deletes made by this connection for synchronization.
public void	StopSynchronizationDelete() [page 156]	UL Ext: Prevents delete operations from being synchronized.
public void	Synchronize [page 157]	UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.
public unsafe int	TriggerEvent(string, string) [page 159]	UL Ext: Triggers an event.
public void	ValidateDatabase [page 160]	UL Ext: Performs validation on the current database.

Properties

Modifier and Type	Property	Description
public override string	ConnectionString [page 163]	Specifies the parameters to use for opening a connection to an UltraLite.NET database.
public override int	ConnectionTimeout [page 164]	This feature is not supported by UltraLite.NET.
public override string	Database [page 165]	Returns the name of the database to which the connection opens.
public unsafe long	DatabaseID [page 165]	UL Ext: Specifies the Database ID value to be used for global autoincrement columns.
public override string	DataSource [page 166]	This feature is not supported by UltraLite.NET.
public unsafe short	GlobalAutoIncrementUsage [page 166]	UL Ext: Returns the percentage of available global autoincrement values that have been used.
public unsafe ulong	LastIdentity [page 167]	UL Ext: Returns the most recent identity value used.
public ULDatabaseSchema	Schema [page 168]	UL Ext: Provides access to the schema of the current database associated with this connection.
public override string	ServerVersion [page 168]	This feature is not supported by UltraLite.NET.
public override ConnectionState	State [page 169]	Returns the current state of the connection.
public ULSyncParms	SyncParms [page 169]	UL Ext: Specifies the synchronization settings for this connection.
public ULSyncResult	SyncResult [page 170]	UL Ext: Returns the results of the last synchronization for this connection.

Events

Modifier and Type	Event	Description
public ULInfoMessageEventHandler	InfoMessage [page 170]	Occurs when UltraLite.NET sends a warning or an informational message on this connection.
public override StateChangeEventHandler	StateChange [page 172]	Occurs when this connection changes state.

Remarks

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

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

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 value.

In this section:

ULConnection Constructor [page 115]
Initializes a ULConnection object.

BeginDbTransaction(IsolationLevel) Method [page 117]

BeginSynchronize Method [page 118]
UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.

BeginTransaction Method [page 121]
Returns a transaction object.

CancelGetNotification(string) Method [page 124]
UL Ext: Cancels any pending get-notification calls on all queues matching the given name.

CancelSynchronize(IAsyncResult) Method [page 125]
UL Ext: Causes a running synchronization to be canceled at the next opportunity.

ChangeDatabase(string) Method [page 125]
Changes the current database for an open connection.

ChangeEncryptionKey(string) Method [page 126]
UL Ext: Changes the database's encryption key to the specified new key.

ChangePassword(string, string) Method [page 127]
Changes the password for the user indicated in the connection string to the supplied new password.

Close() Method [page 128]
Closes the database connection.

CountUploadRows(string, long) Method [page 128]
UL Ext: Returns the number of rows that need to be uploaded when the next synchronization takes place.

Specifies the listener object used to process synchronization messages.

StartSynchronizationDelete() Method [page 155]
UL Ext: Marks all subsequent deletes made by this connection for synchronization.

StopSynchronizationDelete() Method [page 156]
UL Ext: Prevents delete operations from being synchronized.

Synchronize Method [page 157]
UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.

TriggerEvent(string, string) Method [page 159]
UL Ext: Triggers an event.

ValidateDatabase Method [page 160]
UL Ext: Performs validation on the current database.

ConnectionString Property [page 163]
Specifies the parameters to use for opening a connection to an UltraLite.NET database.

ConnectionTimeout Property [page 164]
This feature is not supported by UltraLite.NET.

Database Property [page 165]
Returns the name of the database to which the connection opens.

DatabaseID Property [page 165]
UL Ext: Specifies the Database ID value to be used for global autoincrement columns.

DataSource Property [page 166]
This feature is not supported by UltraLite.NET.

GlobalAutoIncrementUsage Property [page 166]
UL Ext: Returns the percentage of available global autoincrement values that have been used.

LastIdentity Property [page 167]
UL Ext: Returns the most recent identity value used.

Schema Property [page 168]
UL Ext: Provides access to the schema of the current database associated with this connection.

ServerVersion Property [page 168]
This feature is not supported by UltraLite.NET.

State Property [page 169]
Returns the current state of the connection.

SyncParms Property [page 169]
UL Ext: Specifies the synchronization settings for this connection.

SyncResult Property [page 170]
UL Ext: Returns the results of the last synchronization for this connection.

InfoMessage Event [page 170]
Occurs when UltraLite.NET sends a warning or an informational message on this connection.

StateChange Event [page 172]
Occurs when this connection changes state.

Related Information

Open() Method [page 149]
Schema Property [page 168]

1.1.7.1 ULConnection Constructor

Initializes a ULConnection object.

Overload List

Modifier and Type	Overload name	Description
public	ULConnection() [page 115]	Initializes a ULConnection object.
public	ULConnection(string) [page 116]	Initializes a ULConnection object with the specified connection string.

In this section:

ULConnection() Constructor [page 115]
Initializes a ULConnection object.

ULConnection(string) Constructor [page 116]
Initializes a ULConnection object with the specified connection string.

1.1.7.1.1 ULConnection() Constructor

Initializes a ULConnection object.

Syntax

Visual Basic

```
Public Sub ULConnection ()
```

C#

```
public ULConnection ()
```

Remarks

The connection must be opened before you can perform any operations against the database.

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

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

Related Information

Open() Method [page 149]
ConnectionString Property [page 163]

1.1.7.1.2 ULConnection(string) Constructor

Initializes a ULConnection object with the specified connection string.

Syntax

Visual Basic

```
Public Sub ULConnection (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.

Exceptions

ArgumentException The supplied connection string is invalid.

Remarks

The connection must be opened before you can perform any operations against the database.

To use the UltraLite Engine runtime of UltraLite.NET, set the ULDatabaseManager.RuntimeType property 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 Mobile device.

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

The following code is the C# language equivalent:

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

Related Information

Open() Method [page 149]
ULConnection() Constructor [page 115]
ULConnectionParms Class [page 173]
ConnectionString Property [page 163]

1.1.7.2 BeginDbTransaction(IsolationLevel) Method

Syntax

Visual Basic

```
Protected Overrides Function BeginDbTransaction (ByVal isolationLevel As
IsolationLevel) As DbTransaction
```

C#

```
protected override DbTransaction BeginDbTransaction (IsolationLevel
isolationLevel)
```

1.1.7.3 BeginSynchronize Method

UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.

Overload List

Modifier and Type	Overload name	Description
public IAsyncResult	BeginSynchronize() [page 118]	UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.
public IAsyncResult	BeginSynchronize(Control, ULSyncProgressedDlg, object) [page 119]	UL Ext: Asynchronously launches a synchronization using the current SyncParms.

In this section:

BeginSynchronize() Method [page 118]
UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.

BeginSynchronize(Control, ULSyncProgressedDlg, object) Method [page 119]
UL Ext: Asynchronously launches a synchronization using the current SyncParms.

1.1.7.3.1 BeginSynchronize() Method

UL Ext: Asynchronously launches a synchronization using the current the SyncParms object.

Syntax

Visual Basic

```
Public Function BeginSynchronize () As IAsyncResult
```

C#

```
public IAsyncResult BeginSynchronize ()
```

Returns

An IAsyncResult object that can be used to determine if the sync has completed or block until the sync has finished.

Exceptions

ULException class A SQL error occurred.

Remarks

This method will create a new thread to do the synchronization and then return immediately. Call the EndSynchronize method to block until the sync has completed.

Related Information

BeginSynchronize(Control, ULSyncProgressedDlg, object) Method [page 119]
EndSynchronize(IAsyncResult) Method [page 133]
CancelSynchronize(IAsyncResult) Method [page 125]
ULSyncProgressedDlg(IAsyncResult, ULSyncProgressData) Delegate [page 565]

1.1.7.3.2 BeginSynchronize(Control, ULSyncProgressedDlg, object) Method

UL Ext: Asynchronously launches a synchronization using the current SyncParms.

Syntax

Visual Basic

```
Public Function BeginSynchronize (
    ByVal control As Control,
    ByVal dlg As ULSyncProgressedDlg,
    ByVal state As Object
) As IAsyncResult
```

C#

```
public IAsyncResult BeginSynchronize (
    Control control,
    ULSyncProgressedDlg dlg,
    object state
```

```
)
```

Parameters

control A System.Windows.Forms.Control object the synchronization thread will use to invoke ULSyncProgressedDlg calls.

dlg A ULSyncProgressedDlg method that will be invoked regularly with synchronization progress updates.

state This user context can be accessed in the ULSyncProgressedDlg method using IAsyncResult.AsyncState.

Returns

An IAsyncResult object that can be used to determine if the sync has completed or block until the sync has finished.

Exceptions

ULException class A SQL error occurred.

Remarks

This method creates a new thread to do the synchronization and then return immediately, invoking the provided ULSyncProgressedDlg method regularly with synchronization progress updates. Call EndSynchronize to block until the sync has completed.

Related Information

BeginSynchronize() Method [page 118]
EndSynchronize(IAsyncResult) Method [page 133]
CancelSynchronize(IAsyncResult) Method [page 125]
ULSyncProgressedDlg(IAsyncResult, ULSyncProgressData) Delegate [page 565]

1.1.7.4 BeginTransaction Method

Returns a transaction object.

Overload List

Modifier and Type	Overload name	Description
public new ULTransaction	BeginTransaction() [page 121]	Returns a transaction object.
public new ULTransaction	BeginTransaction(IsolationLevel) [page 122]	Returns a transaction object with the specified isolation level.

In this section:

BeginTransaction() Method [page 121]
Returns a transaction object.

BeginTransaction(IsolationLevel) Method [page 122]
Returns a transaction object with the specified isolation level.

1.1.7.4.1 BeginTransaction() Method

Returns a transaction object.

Syntax

Visual Basic

```
Public Shadows Function BeginTransaction () As ULTransaction
```

C#

```
public new ULTransaction BeginTransaction ()
```

Returns

A ULTransaction object representing the new transaction.

Exceptions

ULException class The connection is closed.

InvalidOperationException The ULConnection class does not support parallel transactions.

Remarks

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

The transaction is created with the IsolationLevel.ReadCommitted value.

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

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 and the previous isolation level until the next call to BeginTransaction method.

UltraLite's definition of each isolation level is slightly different than ADO.NET's documentation of IsolationLevel.

This is the strongly-typed version of the System.Data.IDbConnection.BeginTransaction and System.Data.Common.DbConnection.BeginTransaction() methods.

Related Information

Commit() Method [page 559]
Rollback() Method [page 560]
Transaction Property [page 91]
CreateCommand() Method [page 129]

1.1.7.4.2 BeginTransaction(IsolationLevel) Method

Returns a transaction object with the specified isolation level.

Syntax

Visual Basic

```
Public Shadows Function BeginTransaction (ByVal isolationLevel As
IsolationLevel) As ULTransaction
```

C#

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

Parameters

isolationLevel The required isolation level for the transaction. UltraLite.NET only supports the System.Data.IsolationLevel.ReadUncommitted and ReadCommitted values.

Returns

A ULTransaction object representing the new transaction.

Exceptions

ULException class The connection is closed or an unsupported isolation level was specified.
InvalidOperationException The ULConnection class does not support parallel transactions.

Remarks

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

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

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 and the previous isolation level until the next call to the BeginTransaction method.

UltraLite's definition of each isolation level is slightly different than ADO.NET's documentation of IsolationLevel.

This is the strongly-typed version of the System.Data.IDbConnection.BeginTransaction(System.Data.IsolationLevel) and System.Data.Common.DbConnection.BeginTransaction(System.Data.IsolationLevel) methods.

Related Information

ULTransaction Class [page 557]
BeginTransaction() Method [page 121]
Commit() Method [page 559]
Rollback() Method [page 560]
Transaction Property [page 91]
CreateCommand() Method [page 129]

1.1.7.5 CancelGetNotification(string) Method

UL Ext: Cancels any pending get-notification calls on all queues matching the given name.

Syntax

Visual Basic

```
Public Function CancelGetNotification (ByVal queueName As String) As
Integer
```

C#

```
public unsafe int CancelGetNotification (string queueName)
```

Parameters

queueName The expression to match queue names upon.

Returns

The number of affected queues (not the number of blocked reads necessarily).

Exceptions

ULException class A SQL error occurred.

Remarks

This method cancels any pending get-notification calls on all queues matching the given name.

Related Information

GetNotification(string, int) Method [page 142]
ULException Class [page 312]

1.1.7.6 CancelSynchronize(IAsyncResult) Method

UL Ext: Causes a running synchronization to be canceled at the next opportunity.

Syntax

Visual Basic

```
Public Sub CancelSynchronize (ByVal asyncResult As IAsyncResult)
```

C#

```
public void CancelSynchronize (IAsyncResult asyncResult)
```

Parameters

asyncResult The IAsyncResult returned from the BeginSynchronize method.

Remarks

This method will inform the synchronization thread to terminate and returns immediately. Call the EndSynchronize method to block until the sync has successfully terminated.

Related Information

BeginSynchronize() Method [page 118]
BeginSynchronize(Control, ULSyncProgressedDlg, object) Method [page 119]
EndSynchronize(IAsyncResult) Method [page 133]

1.1.7.7 ChangeDatabase(string) Method

Changes the current database for an open connection.

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 value.

Related Information

ConnectionString Property [page 163]

1.1.7.8 ChangeEncryptionKey(string) 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.

Exceptions

ULException class A SQL error occurred.

Remarks

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

Related Information

EncryptionKey Property [page 181]

1.1.7.9 ChangePassword(string, string) 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.

Exceptions

ArgumentNullException Either the connectionString or newPassword parameter is null.

ArgumentException The connection string includes the option to use integrated security.

ULException class A SQL error occurred while attempting to open the database.

1.1.7.10 Close() Method

Closes the database connection.

Syntax

Visual Basic

```
Public Overrides Sub Close ()
```

C#

```
public override void Close ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

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

1.1.7.11 CountUploadRows(string, long) 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 pubs As String,
    ByVal threshold As Long
) As Long
```

C#

```
public unsafe long CountUploadRows (
    string pubs,
    long threshold
)
```

Parameters

pubs A comma separated list of publications to check for rows.
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.

Returns

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

Exceptions

ULException class A SQL error occurred.

1.1.7.12 CreateCommand() Method

Creates and initializes a ULCommand object associated with this connection and its current transaction.

Syntax

Visual Basic

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

C#

```
public new ULCommand CreateCommand ()
```

Returns

A new ULCommand object.

Remarks

You can use the properties of the ULCommand object to control its behavior.

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

This is the strongly-typed version of the System.Data.IDbConnection.CreateCommand and System.Data.Common.DbConnection.CreateCommand methods.

Related Information

ULCommand Class [page 45]
CommandText Property [page 85]

1.1.7.13 CreateDbCommand() Method

Syntax

Visual Basic

```
Protected Overrides Function CreateDbCommand () As DbCommand
```

C#

```
protected override DbCommand CreateDbCommand ()
```

1.1.7.14 CreateNotificationQueue(string, string) Method

UL Ext: Creates an event queue.

Syntax

Visual Basic

```
Public Sub CreateNotificationQueue (
    ByVal queueName As String,
    ByVal parameters As String
)
```

C#

```
public void CreateNotificationQueue (
    string queueName,
    string parameters
)
```

Parameters

queueName The name of the new queue.

parameters Creation parameters; currently unused, set to NULL.

Exceptions

ULException class A SQL error occurred.

Remarks

This method create an event notification queue for this connection. Queue names are scoped per-connection, so different connections can create queues with the same name. When an event notification is sent, all queues in the database with a matching name receive (a separate instance of) the notification. Names are case insensitive. A default queue is created on demand for each connection when calling the RegisterForEvent event if no queue is specified.

Related Information

DestroyNotificationQueue(string) Method [page 132]
ULException Class [page 312]

1.1.7.15 DeclareEvent(string) Method

UL Ext: Declares a named event.

Syntax

Visual Basic

```
Public Sub DeclareEvent (ByVal eventName As String)
```

C#

```
public void DeclareEvent (string eventName)
```

Parameters

eventName The event name.

Exceptions

ULException class A SQL error occurred.

Remarks

Declare an event which can then be registered for and triggered. UltraLite predefines some system events triggered by operations on the database or the environment. The event name must be unique. Names are case insensitive. Throws error if name already used or invalid

Related Information

CreateNotificationQueue(string, string) Method [page 130]
ULException Class [page 312]

1.1.7.16 DestroyNotificationQueue(string) Method

UL Ext: Destroys an event queue.

Syntax

Visual Basic

```
Public Sub DestroyNotificationQueue (ByVal queueName As String)
```

C#

```
public void DestroyNotificationQueue (string queueName)
```

Parameters

queueName The name of the queue.

Exceptions

ULException class A SQL error occurred.

Remarks

Destroy the given event notification queue. A warning is signaled if unread notifications remain in the queue. Unread notifications are discarded. A connection's default event queue, if created, is destroyed when the connection is closed.

Related Information

CreateNotificationQueue(string, string) Method [page 130]
ULException Class [page 312]

1.1.7.17 Dispose(bool) Method

Releases the unmanaged resources used by the ULCommand object and optionally releases the managed resources.

Syntax

Visual Basic

```
Protected Overrides Sub Dispose (ByVal disposing As Boolean)
```

C#

```
protected override unsafe void Dispose (bool disposing)
```

Parameters

disposing When true, dispose of both managed and unmanaged resources. When false, dispose of only the unmanaged resources.

1.1.7.18 EndSynchronize(IAsyncResult) Method

UL Ext: Blocks until an asynchronously launched synchronization terminates.

Syntax

Visual Basic

```
Public Sub EndSynchronize (ByVal asyncResult As IAsyncResult)
```

C#

```
public void EndSynchronize (IAsyncResult asyncResult)
```

Parameters

asyncResult The IAsyncResult returned from the BeginSynchronize method.

Exceptions

ULException class A SQL error occurred.

Remarks

If an error occurred during the synchronization, a ULException exception is thrown.

Related Information

BeginSynchronize() Method [page 118]
BeginSynchronize(Control, ULSyncProgressedDlg, object) Method [page 119]
CancelSynchronize(IAsyncResult) Method [page 125]

1.1.7.19 ExecuteTable Method

UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

Overload List

Modifier and Type	Overload name	Description
public ULTable	ExecuteTable(string) [page 135]	UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

Modifier and Type	Overload name	Description
public ULTable	ExecuteTable(string, string) [page 137]	UL Ext: Retrieves a database table in a ULTable object for direct manipulation.
public ULTable	ExecuteTable(string, string, CommandBehavior) [page 138]	UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

In this section:

ExecuteTable(string) Method [page 135]
UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

ExecuteTable(string, string) Method [page 137]
UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

ExecuteTable(string, string, CommandBehavior) Method [page 138]
UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

1.1.7.19.1 ExecuteTable(string) Method

UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

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.

Returns

The table as a ULTable object.

Exceptions

ULException class A SQL error occurred.
InvalidOperationException The *tableName* is invalid.

Remarks

The table is opened (sorted) using the table's primary key.

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

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()
```

The following code is the C# language equivalent:

```
// 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();
// }
```

Related Information

ExecuteTable(string, string) Method [page 137]
ULTable Class [page 507]
ExecuteTable() Method [page 81]
ULCommand Class [page 45]

1.1.7.19.2 ExecuteTable(string, string) Method

UL Ext: Retrieves a database table in a ULTable object for direct manipulation.

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.

Returns

The table as a ULTable object.

Exceptions

ULException class A SQL error occurred.
InvalidOperationException The *tableName* is invalid.

Remarks

The table is opened (sorted) using the specified index.

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

Example

The following code opens the table named MyTable using the index named MyIndex. It assumes an open ULConnection object 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()
```

The following code is the C# language equivalent:

```
// 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();
// }
```

Related Information

ExecuteTable(string) Method [page 135]
ULTable Class [page 507]
ExecuteTable() Method [page 81]
ULCommand Class [page 45]

1.1.7.19.3 ExecuteTable(string, string, CommandBehavior) Method

UL Ext: Retrieves, with the specified command behavior, a database table for direct manipulation.

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.

Returns

The table as a ULTable object.

Exceptions

ULException class A SQL error occurred.
InvalidOperationException The *tableName* is invalid.

Remarks

The table is opened (sorted) using the specified index.

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

Example

The following code opens the table named MyTable using the index named MyIndex. It assumes an open ULConnection object named conn.

```
' Visual Basic
Dim t As ULTable = conn.ExecuteTable( _
   "MyTable", "MyIndex", CommandBehavior.Default _
)
' The line above is equivalent to the following code:
' 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()
```

The following code is the C# language equivalent:

```
// C#
ULTable t = conn.ExecuteTable(
   "MyTable", "MyIndex", CommandBehavior.Default
);
// The line above is equivalent to the following code:
// ULTable t;
// using(ULCommand cmd = conn.CreateCommand())
// {
//     cmd.CommandText = "MyTable";
//     cmd.IndexName = "MyIndex";
//     cmd.CommandType = CommandType.TableDirect;
//     t = cmd.ExecuteTable(CommandBehavior.Default);
// }
```

Related Information

ExecuteTable(string) Method [page 135]
ExecuteTable(string, string) Method [page 137]
ULTable Class [page 507]
ULCommand Class [page 45]

1.1.7.20 GetLastDownloadTime(string) Method

UL Ext: Returns the time of the most recent download of the specified publication.

Syntax

Visual Basic

```
Public Function GetLastDownloadTime (ByVal publication As String) As Date
```

C#

```
public unsafe DateTime GetLastDownloadTime (string publication)
```

Parameters

publication The publication to check.

Returns

The timestamp of the last download. If the SYNC_ALL_DB constant is used for publication, returns the time of the last download of the entire database.

Exceptions

ULException class A SQL error occurred.

Remarks

The parameter *publication* is a publication name to check.

Related Information

ResetLastDownloadTime(string) Method [page 151]

1.1.7.21 GetNewUUID() Method

UL Ext: Generates a new UUID (System.Guid).

Syntax

Visual Basic

```
Public Function GetNewUUID () As Guid
```

C#

```
public unsafe Guid GetNewUUID ()
```

Returns

A new UUID as a System.Guid.

Exceptions

ULException class A SQL error occurred.

Remarks

This method is provided here because it is not included in the .NET Compact Framework.

1.1.7.22 GetNotification(string, int) Method

UL Ext: Blocks for a notification or timeout.

Syntax

Visual Basic

```
Public Function GetNotification (
    ByVal queueName As String,
    ByVal wait_ms As Integer
) As String
```

C#

```
public string GetNotification (
    string queueName,
    int wait_ms
)
```

Parameters

queueName The name of the queue to be waited upon.

wait_ms The time to wait, in milliseconds. Use System.Threading.Timeout.Infinite (-1) for an indefinite wait.

Returns

Null if wait period expired or was canceled; otherwise, returns the event name.

Exceptions

ULException class A SQL error occurred.

Remarks

This method reads an event notification. This call blocks until a notification is received or until the given wait period expires. To wait indefinitely, pass System.Threading.Timeout.Infinite for the wait_ms parameter. To cancel a wait, send another notification to the given queue or use the CancelGetNotification method. After reading a notification, use the ReadNotificationParameter method to retrieve additional parameters.

Related Information

SendNotification(string, string, string) Method [page 153]
GetNotificationParameter(string, string) Method [page 143]
CancelGetNotification(string) Method [page 124]
ULException Class [page 312]

1.1.7.23 GetNotificationParameter(string, string) Method

UL Ext: Gets the value of a parameter for an event that was just read by the GetNotification method.

Syntax

Visual Basic

```
Public Function GetNotificationParameter (
    ByVal queueName As String,
    ByVal parameterName As String
) As String
```

C#

```
public string GetNotificationParameter (
```

```
    string queueName,
    string parameterName
)
```

Parameters

queueName The name of the queue to be waited upon.
parameterName The name of the parameter whose value should be returned.

Returns

The parameter value if the parameter was found; otherwise, returns null.

Exceptions

ULException class A SQL error occurred.

Remarks

This method get the value of a parameter for the event notification just read by the ULGetNotification method. Only the parameters from the most-recently read notification on the given queue are available.

Related Information

GetNotification(string, int) Method [page 142]
ULException Class [page 312]

1.1.7.24 GetSchema Method

Returns the list of supported schema collections.

Overload List

Modifier and Type	Overload name	Description
public override DataTable	GetSchema() [page 145]	Returns the list of supported schema collections.
public override DataTable	GetSchema(string) [page 146]	Returns information for the specified metadata collection for this ULConnection object.
public override DataTable	GetSchema(string, string[]) [page 146]	Returns schema information for the data source of this ULConnection object and, if specified, uses the specified string for the schema name and the specified string array for the restriction values.

In this section:

GetSchema() Method [page 145]
Returns the list of supported schema collections.

GetSchema(string) Method [page 146]
Returns information for the specified metadata collection for this ULConnection object.

GetSchema(string, string[]) Method [page 146]
Returns schema information for the data source of this ULConnection object and, if specified, uses the specified string for the schema name and the specified string array for the restriction values.

1.1.7.24.1 GetSchema() Method

Returns the list of supported schema collections.

Syntax

Visual Basic

```
Public Overrides Function GetSchema () As DataTable
```

C#

```
public override DataTable GetSchema ()
```

1.1.7.24.2 GetSchema(string) Method

Returns information for the specified metadata collection for this ULConnection object.

Syntax

Visual Basic

```
Public Overrides Function GetSchema (ByVal collection As String) As
DataTable
```

C#

```
public override DataTable GetSchema (string collection)
```

Parameters

collection The name of the metadata collection. If no name is provided, the MetaDataCollections value is used.

Related Information

GetSchema(string, string[]) Method [page 146]
ULConnection Class [page 108]

1.1.7.24.3 GetSchema(string, string[]) Method

Returns schema information for the data source of this ULConnection object 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 The name of the metadata collection. If no name is provided, the MetaDataCollections is used.
restrictions A set of restriction values for the requested schema.

Returns

A DataTable object 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 returns 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 about all columns in the database.
DataSourceInformation	Returns information about the database provider.
DataTypes	Returns a list of supported data types.
ForeignKeys	Returns information about all foreign keys in the database.
IndexColumns	Returns information about all index columns in the database.
Indexes	Returns information about all indexes in the database.
MetaDataCollections	Returns a list of all collection names.
Publications	Returns information about all publications in the database.
ReservedWords	Returns a list of reserved words used by UltraLite.
Restrictions	Returns information about restrictions used in GetSchema.
Tables	Returns information about 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 the GetSchema method.

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 about all tables named my_table.

GetSchema("Tables", new string[ ] { NULL, "User", NULL }) Returns information about all user tables.

Related Information

ULConnection Class [page 108]
ULMetaDataCollectionNames Class [page 359]

1.1.7.25 GrantConnectTo(string, string) 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.
pwd The password to be associated with the user ID.

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.

Related Information

UserID Property [page 182]
Password Property [page 181]
ConnectionString Property [page 163]

1.1.7.26 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 ()
```

Exceptions

InvalidOperationException The connection is already open or the connection string is not specified in the ULConnection.ConnectionString property.

ULException class A SQL error occurred while attempting to open the database.

Remarks

You should explicitly close or dispose of the connection when you are done with it.

Related Information

ConnectionString Property [page 163]
State Property [page 169]

1.1.7.27 RegisterForEvent(string, string, string, bool) Method

UL Ext: Registers a queue to get events from an object.

Syntax

Visual Basic

```
Public Sub RegisterForEvent (
    ByVal eventName As String,
    ByVal objectName As String,
    ByVal queueName As String,
    ByVal registerNotUnReg As Boolean
)
```

C#

```
public void RegisterForEvent (
    string eventName,
    string objectName,
    string queueName,
    bool registerNotUnReg
)
```

Parameters

eventName The event name.
objectName The object name to which event applies. For example, a table name.
queueName The event queue name to be used.
registerNotUnReg True to register; false to unregister.

Exceptions

ULException class A SQL error occurred.

Remarks

This method registers a queue to receive notifications of an event. The default connection queue is implied and created if a queue name is not supplied. Certain system events allow specification of an object name to which

the event applies. For example, the TableModified event can specify the table name. Unlike the SendNotification method, only the specific queue registered receives notifications of the event; other queues with the same name on different connections do not (unless they are also explicit registered). This method throws an error if the queue or event does not exist.

Related Information

DeclareEvent(string) Method [page 131]
CreateNotificationQueue(string, string) Method [page 130]
ULException Class [page 312]

1.1.7.28 ResetLastDownloadTime(string) Method

UL Ext: Resets the time of the most recent download.

Syntax

Visual Basic

```
Public Sub ResetLastDownloadTime (ByVal pubs As String)
```

C#

```
public void ResetLastDownloadTime (string pubs)
```

Exceptions

ULException class A SQL error occurred.

Related Information

GetLastDownloadTime(string) Method [page 140]

1.1.7.29 RevokeConnectFrom(string) 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.

Related Information

GrantConnectTo(string, string) Method [page 148]

1.1.7.30 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.

Related Information

KeepPartialDownload Property [page 475]
ResumePartialDownload Property [page 479]

1.1.7.31 SendNotification(string, string, string) Method

UL Ext: Sends a notification to matching queues.

Syntax

Visual Basic

```
Public Function SendNotification (
    ByVal queueName As String,
    ByVal eventName As String,
    ByVal parameters As String
) As Integer
```

C#

```
public unsafe int SendNotification (
    string queueName,
    string eventName,
    string parameters
)
```

Parameters

queueName The event queue name to be used.
eventName The event name.
parameters Parameters to pass.

Returns

The number of notifications sent (the number of matching queues).

Exceptions

ULException class A SQL error occurred.

Remarks

Returns the number of matching queues.

This method send a notification to all queues matching the given name (including any such queue on the current connection). This call does not block. Use the special queue name "*" to send to all queues.

Related Information

DeclareEvent(string) Method [page 131]
RegisterForEvent(string, string, string, bool) Method [page 150]
ULException Class [page 312]

1.1.7.32 SetSyncListener(ULSyncProgressListener) Method

Specifies the listener object used to process synchronization messages.

Syntax

Visual Basic

```
Public Sub SetSyncListener (ByVal listener As ULSyncProgressListener)
```

C#

```
public void SetSyncListener (ULSyncProgressListener listener)
```

Parameters

listener The ULSyncProgressListener object that implements the SyncProgressed method, which is called for synchronization messages on this connection.

Exceptions

ULException class A SQL error occurred.

Remarks

When the SYNCHRONIZE profileName SQL statement is executed, its progress messages are routed to a syncListener object, if not null (Nothing in Visual Basic).

To remove the listener, pass a null reference in a call to the SetSyncListener method.

Related Information

ULSyncProgressListener Interface [page 499]

1.1.7.33 StartSynchronizationDelete() Method

UL Ext: Marks all subsequent deletes made by this connection for synchronization.

Syntax

Visual Basic

```
Public Sub StartSynchronizationDelete ()
```

C#

```
public void StartSynchronizationDelete ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

When this method 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.

Related Information

StopSynchronizationDelete() Method [page 156]
Truncate() Method [page 535]

1.1.7.34 StopSynchronizationDelete() Method

UL Ext: Prevents delete operations from being synchronized.

Syntax

Visual Basic

```
Public Sub StopSynchronizationDelete ()
```

C#

```
public void StopSynchronizationDelete ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

This method is useful for deleting old information about an UltraLite database to save space, while not deleting this information about the consolidated database.

Related Information

StartSynchronizationDelete() Method [page 155]

1.1.7.35 Synchronize Method

UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.

Overload List

Modifier and Type	Overload name	Description
public void	Synchronize() [page 157]	UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.
public void	Synchronize(ULSyncProgressListener) [page 158]	UL Ext: Synchronizes the database using the current ULConnection.SyncParms object with progress events posted to the specified listener.

In this section:

Synchronize() Method [page 157]
UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.

Synchronize(ULSyncProgressListener) Method [page 158]
UL Ext: Synchronizes the database using the current ULConnection.SyncParms object with progress events posted to the specified listener.

1.1.7.35.1 Synchronize() Method

UL Ext: Synchronizes the database using the current ULConnection.SyncParms object.

Syntax

Visual Basic

```
Public Sub Synchronize ()
```

C#

```
public void Synchronize ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

A detailed result status is reported in this connection's ULConnection.SyncResult property.

Related Information

Synchronize(ULSyncProgressListener) Method [page 158]
SyncParms Property [page 169]
SyncResult Property [page 170]

1.1.7.35.2 Synchronize(ULSyncProgressListener) Method

UL Ext: Synchronizes the database using the current ULConnection.SyncParms object 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.

Exceptions

ULException class A SQL error occurred.

Remarks

Errors during synchronization are posted as a ULSyncProgressState.STATE_ERROR event, then thrown as a ULException.

A detailed result status is reported in this connection's ULConnection.SyncResult property.

Related Information

ULSyncProgressListener Interface [page 499]
Synchronize() Method [page 157]
SyncParms Property [page 169]
ULException Class [page 312]
SyncResult Property [page 170]

1.1.7.36 TriggerEvent(string, string) Method

UL Ext: Triggers an event.

Syntax

Visual Basic

```
Public Function TriggerEvent (
    ByVal eventName As String,
    ByVal parameters As String
) As Integer
```

C#

```
public unsafe int TriggerEvent (
    string eventName,
    string parameters
)
```

Parameters

eventName The event name to be triggered.
parameters Parameters to pass.

Returns

The number of event notifications sent.

Exceptions

ULException class A SQL error occurred.

Remarks

Returns the number of notifications sent.

This method triggers an event (and send notification to all registered queues).

Related Information

DeclareEvent(string) Method [page 131]
RegisterForEvent(string, string, string, bool) Method [page 150]
ULException Class [page 312]

1.1.7.37 ValidateDatabase Method

UL Ext: Performs validation on the current database.

Overload List

Modifier and Type	Overload name	Description
public void	ValidateDatabase(ULDBValid) [page 161]	UL Ext: Performs validation on the current database.
public void	ValidateDatabase(ULDBValid, string) [page 162]	UL Ext: Performs validation on the current database.

In this section:

ValidateDatabase(ULDBValid) Method [page 161]
UL Ext: Performs validation on the current database.

ValidateDatabase(ULDBValid, string) Method [page 162]
UL Ext: Performs validation on the current database.

1.1.7.37.1 ValidateDatabase(ULDBValid) Method

UL Ext: Performs validation on the current database.

Syntax

Visual Basic

```
Public Sub ValidateDatabase (ByVal how As ULDBValid)
```

C#

```
public void ValidateDatabase (ULDBValid how)
```

Parameters

how Describes how to validate the database.

Exceptions

ULException class A SQLE_CORRUPT_ULTRALITE_INDEX or SQLE_CORRUPT_ULTRALITE_DATABASE error may occur if the database is corrupt.

Example

The following code validates the current database

```
' Visual Basic
conn.ValidateDatabase( Sap.Data.UltraLite.ULVF_INDEX )
```

The following code is the C# language equivalent:

```
// C#
conn.ValidateDatabase( Sap.Data.UltraLite.ULVF_INDEX )
```

Related Information

ValidateDatabase(string, ULDBValid) Method [page 247]
ULDBValid Enumeration [page 571]

1.1.7.37.2 ValidateDatabase(ULDBValid, string) Method

UL Ext: Performs validation on the current database.

Syntax

Visual Basic

```
Public Sub ValidateDatabase (
    ByVal how As ULDBValid,
    ByVal tableName As String
)
```

C#

```
public void ValidateDatabase (
    ULDBValid how,
    string tableName
)
```

Parameters

how Describes how to validate the database.

tableName If null (Nothing in Visual Basic), validate the entire database; otherwise, validate just the named table.

Exceptions

ULException class A SQLE_CORRUPT_ULTRALITE_INDEX or SQLE_CORRUPT_ULTRALITE_DATABASE error may occur if the database is corrupt.

Example

The following code validates the current database

```
' Visual Basic
conn.ValidateDatabase( Sap.Data.UltraLite.ULVF_INDEX, Nothing )
```

The following code is the C# language equivalent:

```
// C#
conn.ValidateDatabase( Sap.Data.UltraLite.ULVF_INDEX, null )
```

Related Information

ValidateDatabase(string, ULDBValid) Method [page 247]
ULDBValid Enumeration [page 571]

1.1.7.38 ConnectionString Property

Specifies the parameters to use for opening a connection to an UltraLite.NET database.

Syntax

Visual Basic

```
Public Overrides Property ConnectionString As String
```

C#

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

Remarks

The connection string can be supplied using a ULConnectionParms object.

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

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.

Parameter values can be quoted with either single quote characters or double quote characters provided that the quoted contents do not contain quote characters of the same type. Values must be quoted if they contain semicolons, begin with a quote, or require leading or trailing whitespace.

If you are not quoting parameter values, make sure that they do not contain semicolons, and that they 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 the GrantConnectTo method) 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 Mobile device.

```
' Visual Basic
```

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

The following code is the C# language equivalent:

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

Related Information

Open() Method [page 149]
ULConnectionParms Class [page 173]
GrantConnectTo(string, string) Method [page 148]

1.1.7.39 ConnectionTimeout Property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property ConnectionTimeout As Integer
```

C#

```
public override int ConnectionTimeout {get;}
```

Remarks

The value is always zero.

1.1.7.40 Database Property

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

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Database As String
```

C#

```
public override string Database {get;}
```

Remarks

A string containing the name of the database.

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

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

1.1.7.41 DatabaseID 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 unsafe long DatabaseID {get;set;}
```

Remarks

The Database ID value of the current database.

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.

Related Information

GetDatabaseProperty(string) Method [page 251]
SetDatabaseOption(string, string) Method [page 256]

1.1.7.42 DataSource Property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property DataSource As String
```

C#

```
public override string DataSource {get;}
```

Remarks

The value is always the empty string.

1.1.7.43 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 unsafe short GlobalAutoIncrementUsage {get;}
```

Remarks

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

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

Related Information

ULDatabaseManager Class [page 240]
DatabaseID Property [page 165]

1.1.7.44 LastIdentity Property

UL Ext: Returns the most recent identity value used.

Syntax

Visual Basic

```
Public ReadOnly Property LastIdentity As ULong
```

C#

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

Remarks

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

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

```
SELECT @identity
```

The LastIdentity property 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, the LastIdentity property 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.

1.1.7.45 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;}
```

Remarks

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

This property is only valid while its connection is open.

Related Information

ULDatabaseSchema Class [page 250]

1.1.7.46 ServerVersion Property

This feature is not supported by UltraLite.NET.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property ServerVersion As String
```

C#

```
public override string ServerVersion {get;}
```

Remarks

The value is always the empty string.

1.1.7.47 State Property

Returns the current state of the connection.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property State As ConnectionState
```

C#

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

Remarks

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

Related Information

StateChange Event [page 172]

1.1.7.48 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;}
```

Remarks

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.

Related Information

Synchronize() Method [page 157]
SyncResult Property [page 170]
ULSyncParms Class [page 468]

1.1.7.49 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;}
```

Remarks

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

Related Information

Synchronize() Method [page 157]
SyncParms Property [page 169]

1.1.7.50 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 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
```

The following code is the C# language equivalent:

```
// C#
private void MyInfoMessageHandler(
   object obj, ULInfoMessageEventArgs args
)
{
   System.Console.WriteLine(
       "InfoMessageHandler: " + args.NativeError + ", "
       + args.Message
   );
}
```

The following code adds the MyInfoMessageHandler method to the connection named conn.

```
' Visual Basic
AddHandler conn.InfoMessage, AddressOf MyInfoMessageHandler
```

The following code is the C# language equivalent:

```
// C#
conn.InfoMessage +=
   new ULInfoMessageEventHandler(MyInfoMessageHandler);
```

Related Information

ULInfoMessageEventHandler(object, ULInfoMessageEventArgs) Delegate [page 562]

1.1.7.51 StateChange Event

Occurs when this connection changes state.

Syntax

Visual Basic

```
Public Event StateChange  As StateChangeEventHandler
```

C#

```
public 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
```

The following code is the C# language equivalent:

```
// 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
```

The following code is the C# language equivalent:

```
// C#
conn.StateChange += new StateChangeEventHandler(MyStateHandler);
```

1.1.8 ULConnectionParms Class

UL Ext: Builds a connection string for opening a connection to an UltraLite database.

Syntax

Visual Basic

```
Public Class ULConnectionParms Inherits System.ComponentModel.Component
```

C#

```
public class ULConnectionParms : System.ComponentModel.Component
```

Members

All members of ULConnectionParms, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULConnectionParms() [page 176]	Initializes a ULConnectionParms instance with its default values.

Methods

Modifier and Type	Method	Description
public override string	ToString() [page 176]	Returns the string representation of this instance.

Properties

Modifier and Type	Property	Description
public string	AdditionalParms [page 177]	Specifies additional parameters as a semicolon-separated list of name=value pairs.
public string	CacheSize [page 178]	Specifies the size of the cache.
public string	ConnectionName [page 179]	Specifies a name for the connection.

Modifier and Type	Property	Description
public string	DatabaseOnDesktop [page 180]	Specifies the path and file name of the UltraLite database on Windows desktop platforms.
public string	DatabaseOnDevice [page 180]	Specifies the path and file name of the UltraLite database on Windows Mobile.
public string	EncryptionKey [page 181]	Specifies a key for encrypting the database.
public string	Password [page 181]	Specifies the password for the authenticated user.
public string	UserID [page 182]	Specifies an authenticated user for the database.

Remarks

The frequently-used connection parameters are individual properties on the ULConnectionParms object.

A ULConnectionParms object is used to specify the parameters for opening a connection (with the ULConnection.Open method) or dropping a database (with the ULDatabaseManager.DropDatabase method).

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 method. The resulting string is used to create a new ULConnection object with the ULConnection(String) constructor or set the ULConnection.ConnectionString property 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 Mobile, while mydb2.db would be used on other platforms.

```
' Visual Basic
Dim dbName As ULConnectionParms = new ULConnectionParms
dbName.DatabaseOnDevice = "\UltraLite\mydb1.udb"
dbName.DatabaseOnDesktop = "somedir\mydb2.udb"
```

The following code is the C# language equivalent:

```
// C#
ULConnectionParms dbName = new ULConnectionParms();
dbName.DatabaseOnDevice = "\\UltraLite\\mydb1.udb";
dbName.DatabaseOnDesktop = "somedir\mydb2.udb";
```

The recommended extension for UltraLite database files is .udb. On Windows Mobile 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.

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 value, a non-default ULConnectionParms.Password value, a database ULConnectionParms.EncryptionKey value, and the ULConnectionParms.CacheSize value. If your application is using multiple connections, you should provide a unique ULConnectionParms.ConnectionName value 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 method. To add a user or change a user's password, use the ULConnection.GrantConnectTo method.

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 method.

In this section:

Related Information

Open() Method [page 149]
DropDatabase(string) Method [page 243]

ToString() Method [page 176]
ULConnection Class [page 108]
ULConnection Class [page 108]
ConnectionString Property [page 163]
UserID Property [page 182]
Password Property [page 181]
EncryptionKey Property [page 181]
CacheSize Property [page 178]
ConnectionName Property [page 179]
AdditionalParms Property [page 177]
RevokeConnectFrom(string) Method [page 152]
GrantConnectTo(string, string) Method [page 148]
ChangeEncryptionKey(string) Method [page 126]

1.1.8.1 ULConnectionParms() Constructor

Initializes a ULConnectionParms instance with its default values.

Syntax

Visual Basic

```
Public Sub ULConnectionParms ()
```

C#

```
public ULConnectionParms ()
```

1.1.8.2 ToString() Method

Returns the string representation of this instance.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

Returns

The string representation of this instance as a semicolon-separated list keyword=value pairs.

1.1.8.3 AdditionalParms Property

Specifies additional parameters as a semicolon-separated list of name=value pairs.

Syntax

Visual Basic

```
Public Property AdditionalParms As String
```

C#

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

Remarks

These parameters are used less frequently.

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).

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 value. If one is not found, UltraLite starts a new database using the appropriate database file name parameter (with the DatabaseOnDevice or DatabaseOnDesktop properties). 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 the ULConnection.Open method.

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. The reserve_size parameter reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead and data compression must be considered when deriving the required file system space from the amount of database data. 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 the ULConnection.Open method.
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.

Related Information

RuntimeType Property [page 249]
ConnectionString Property [page 163]
DatabaseOnDevice Property [page 180]
DatabaseOnDesktop Property [page 180]
Open() Method [page 149]

1.1.8.4 CacheSize Property

Specifies the size of the cache.

Syntax

Visual Basic

```
Public Property CacheSize As String
```

C#

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

Remarks

A string specifying the cache size. The default is a null reference (Nothing in Visual Basic) meaning the default of 16 pages is used.

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.

1.1.8.5 ConnectionName Property

Specifies a name for the connection.

Syntax

Visual Basic

```
Public Property ConnectionName As String
```

C#

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

Remarks

This is only needed if you create more than one connection to the database.

A string specifying the name of the connection. The default is a null reference (Nothing in Visual Basic).

1.1.8.6 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;}
```

Remarks

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).

1.1.8.7 DatabaseOnDevice Property

Specifies the path and file name of the UltraLite database on Windows Mobile.

Syntax

Visual Basic

```
Public Property DatabaseOnDevice As String
```

C#

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

Remarks

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).

1.1.8.8 EncryptionKey Property

Specifies a key for encrypting the database.

Syntax

Visual Basic

```
Public Property EncryptionKey As String
```

C#

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

Remarks

A string specifying the encryption key. The default is a null reference (Nothing in Visual Basic) meaning no encryption.

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.

Related Information

ChangeEncryptionKey(string) Method [page 126]

1.1.8.9 Password Property

Specifies the password for the authenticated user.

Syntax

Microsoft Visual Basic

```
Public Property Password As String
```

C#

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

Remarks

A string specifying a database user ID. The default is a null reference (Nothing in Microsoft Visual Basic).

Passwords are case sensitive.

Related Information

UserID Property [page 182]

1.1.8.10 UserID Property

Specifies an authenticated user for the database.

Syntax

Visual Basic

```
Public Property UserID As String
```

C#

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

Remarks

A string specifying a database user ID. The default value is a null reference (Nothing in Visual Basic).

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 (with the ULConnection.GrantConnectTo method) and remove the DBA user (with the ULConnection.RevokeConnectFrom method).

Related Information

Password Property [page 181]
GrantConnectTo(string, string) Method [page 148]
RevokeConnectFrom(string) Method [page 152]

1.1.9 ULConnectionStringBuilder Class

Builds a connection string for opening a connection to an UltraLite database.

Syntax

Visual Basic

```
Public NotInheritable Class ULConnectionStringBuilder Inherits
System.Data.Common.DbConnectionStringBuilder
```

C#

```
public sealed class ULConnectionStringBuilder :
System.Data.Common.DbConnectionStringBuilder
```

Members

All members of ULConnectionStringBuilder, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULConnectionStringBuilder [page 187]	Initializes a ULConnectionStringBuilder object with its default values.

Methods

Modifier and Type	Method	Description
public override bool	ContainsKey(string) [page 188]	Determines whether the ULConnection-StringBuilder object contains a specific keyword.
public override bool	EquivalentTo(DbConnectionString-Builder) [page 189]	Compares the connection information in this ULConnectionStringBuilder object with the connection information in the supplied DbConnectionString-Builder object.
public static string	GetShortName(string) [page 189]	Retrieves the short version of the supplied keyword.
public override bool	Remove(string) [page 190]	Removes the entry with the specified key from the ULConnectionString-Builder object.
public override bool	TryGetValue(string, out Object) [page 190]	Retrieves a value corresponding to the supplied key from this ULConnection-StringBuilder object.

Properties

Modifier and Type	Property	Description
public string	CacheSize [page 191]	UL Ext: Specifies the size of the cache.
public string	ConnectionName [page 192]	Specifies a name for the connection.
public string	DatabaseKey [page 193]	Specifies a key for encrypting the database.
public string	DatabaseName [page 193]	Specifies a name for the database or the name of a loaded database to which a connection needs to be made.
public string	DatabaseOnDesktop [page 194]	UL Ext: Specifies the path and file name of the UltraLite database on Windows desktop platforms.
public string	DatabaseOnDevice [page 195]	UL Ext: Specifies the path and file name of the UltraLite database on Windows Mobile.
public string	OrderedTableScans [page 195]	Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default.
public string	Password [page 196]	Specifies the password for the authenticated user.
public string	ReserveSize [page 196]	UL Ext: Specifies the reserve file system space for storage of UltraLite persistent data.
public string	StartLine [page 197]	Specifies the location and then starts the UltraLite engine.
public override object	this[string keyword] [page 198]	Specifies the value of the specified connection keyword.
public string	UserID [page 199]	Specifies an authenticated user for the database.

Remarks

The frequently-used connection parameters are individual properties on the ULConnectionStringBuilder object.

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 (with the ULConnection.Open method) or dropping a database (with the ULDatabaseManager.DropDatabase method).

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 object with the ULConnection(String) constructor or set the ULConnection.ConnectionString property 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 Mobile, while mydb2.db would be used on other platforms.

```
' Visual Basic
Dim dbName As ULConnectionStringBuilder = _
   new ULConnectionStringBuilder
dbName.DatabaseOnDevice = "\UltraLite\mydb1.udb"
dbName.DatabaseOnDesktop = "somedir\mydb2.udb"
```

The following code is the C# language equivalent:

```
// C#
ULConnectionStringBuilder dbName = new ULConnectionStringBuilder();
dbName.DatabaseOnDevice = "\\UltraLite\\mydb1.udb";
dbName.DatabaseOnDesktop = @"somedir\\mydb2.udb";
```

The recommended extension for UltraLite database files is .udb. On Windows Mobile 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.

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 value, a non-default ULConnectionStringBuilder.Password value, a database ULConnectionStringBuilder.DatabaseKey value, and the ULConnectionStringBuilder.CacheSize value. If your application is using multiple connections, you should provide a unique ULConnectionStringBuilder.ConnectionName value 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, call the ULConnection.RevokeConnectFrom method. To add a user or change a user's password, call the ULConnection.GrantConnectTo method.

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 method.

In this section:

Retrieves the short version of the supplied keyword.

Remove(string) Method [page 190]
Removes the entry with the specified key from the ULConnectionStringBuilder object.

TryGetValue(string, out Object) Method [page 190]
Retrieves a value corresponding to the supplied key from this ULConnectionStringBuilder object.

CacheSize Property [page 191]
UL Ext: Specifies the size of the cache.

ConnectionName Property [page 192]
Specifies a name for the connection.

DatabaseKey Property [page 193]
Specifies a key for encrypting the database.

DatabaseName Property [page 193]
Specifies a name for the database or the name of a loaded database to which a connection needs to be made.

DatabaseOnDesktop Property [page 194]
UL Ext: Specifies the path and file name of the UltraLite database on Windows desktop platforms.

DatabaseOnDevice Property [page 195]
UL Ext: Specifies the path and file name of the UltraLite database on Windows Mobile.

OrderedTableScans Property [page 195]
Specifies whether SQL queries without ORDER BY clauses should perform ordered table scans by default.

Password Property [page 196]
Specifies the password for the authenticated user.

ReserveSize Property [page 196]
UL Ext: Specifies the reserve file system space for storage of UltraLite persistent data.

StartLine Property [page 197]
Specifies the location and then starts the UltraLite engine.

this[string keyword] Property [page 198]
Specifies the value of the specified connection keyword.

UserID Property [page 199]
Specifies an authenticated user for the database.

Related Information

Open() Method [page 149]
DropDatabase(string) Method [page 243]
ULConnection Class [page 108]
ULConnection Class [page 108]
ConnectionString Property [page 163]
DatabaseName Property [page 193]
UserID Property [page 199]

Password Property [page 196]
DatabaseKey Property [page 193]
CacheSize Property [page 191]
ConnectionName Property [page 192]
RevokeConnectFrom(string) Method [page 152]
GrantConnectTo(string, string) Method [page 148]
ChangeEncryptionKey(string) Method [page 126]

1.1.9.1 ULConnectionStringBuilder Constructor

Initializes a ULConnectionStringBuilder object with its default values.

Overload List

Modifier and Type	Overload name	Description
public	ULConnectionStringBuilder() [page 187]	Initializes a ULConnectionStringBuilder object with its default values.
public	ULConnectionStringBuilder(string) [page 188]	Initializes a ULConnectionStringBuilder object with the specified connection string.

In this section:

ULConnectionStringBuilder() Constructor [page 187]
Initializes a ULConnectionStringBuilder object with its default values.

ULConnectionStringBuilder(string) Constructor [page 188]
Initializes a ULConnectionStringBuilder object with the specified connection string.

1.1.9.1.1 ULConnectionStringBuilder() Constructor

Initializes a ULConnectionStringBuilder object with its default values.

Syntax

Visual Basic

```
Public Sub ULConnectionStringBuilder ()
```

C#

```
public ULConnectionStringBuilder ()
```

1.1.9.1.2 ULConnectionStringBuilder(string) Constructor

Initializes a ULConnectionStringBuilder object with the specified connection string.

Syntax

Visual Basic

```
Public Sub ULConnectionStringBuilder (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.

1.1.9.2 ContainsKey(string) 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.

Returns

True if this connection string builder contains a value for the specified keyword, otherwise returns false.

1.1.9.3 EquivalentTo(DbConnectionStringBuilder) 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.

Returns

True if this object is equivalent to the specified DbConnectionStringBuilder object; otherwise, returns false.

1.1.9.4 GetShortName(string) 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.

Returns

The short version of the supplied keyword if keyword is recognized, null otherwise.

1.1.9.5 Remove(string) Method

Removes the entry with the specified key from the ULConnectionStringBuilder object.

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.

Returns

True if the key existed within the connection string and was removed; false if the key did not exist.

1.1.9.6 TryGetValue(string, out Object) Method

Retrieves a value corresponding to the supplied key from this ULConnectionStringBuilder object.

Syntax

Visual Basic

```
Public Overrides Function TryGetValue (
```

```
    ByVal keyword As String,
    ByVal value As Object
) As Boolean
```

C#

```
public override bool TryGetValue (
    string keyword,
    out Object value
)
```

Parameters

keyword The key of the item to retrieve.
value The value corresponding to the key.

Returns

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 the TryGetValue method 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 places the null value (Nothing in Visual Basic) in the value parameter.

1.1.9.7 CacheSize Property

UL Ext: Specifies the size of the cache.

Syntax

Visual Basic

```
Public Property CacheSize As String
```

C#

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

Remarks

A string specifying the cache size. The default is a null reference (Nothing in Visual Basic) meaning the default of 16 pages is used.

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.

1.1.9.8 ConnectionName Property

Specifies a name for the connection.

Syntax

Visual Basic

```
Public Property ConnectionName As String
```

C#

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

Remarks

This is only needed if you create more than one connection to the database.

A string specifying the name of the connection. The default is a null reference (Nothing in Visual Basic).

1.1.9.9 DatabaseKey Property

Specifies a key for encrypting the database.

Syntax

Visual Basic

```
Public Property DatabaseKey As String
```

C#

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

Remarks

A string specifying the encryption key. The default is a null reference (Nothing in Visual Basic) meaning no encryption.

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.

Related Information

ChangeEncryptionKey(string) Method [page 126]

1.1.9.10 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;}
```

Remarks

A string specifying the name of the database. The default is a null reference (Nothing in Visual Basic).

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 parameter. If one is not found, UltraLite starts a new database using the appropriate database file name parameter (the DatabaseOnDevice or DatabaseOnDesktop properties).

This parameter is required if the application (or UltraLite engine) needs to access two different databases that have the same base file name.

Related Information

DatabaseOnDevice Property [page 195]
DatabaseOnDesktop Property [page 194]

1.1.9.11 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;}
```

Remarks

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).

1.1.9.12 DatabaseOnDevice Property

UL Ext: Specifies the path and file name of the UltraLite database on Windows Mobile.

Syntax

Visual Basic

```
Public Property DatabaseOnDevice As String
```

C#

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

Remarks

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).

1.1.9.13 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;}
```

Remarks

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).

When using dynamic SQL in UltraLite, if order is not important for executing a query, UltraLite accesses 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 the OrderedTableScans value 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 defaults to using the primary key.

1.1.9.14 Password Property

Specifies the password for the authenticated user.

Syntax

Visual Basic

```
Public Property Password As String
```

C#

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

Remarks

A string specifying a database user ID. The default is a null reference (Nothing in Visual Basic).

Passwords are case sensitive.

When a database is created, the password for the DBA user ID is set to sql.

Related Information

UserID Property [page 199]

1.1.9.15 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;}
```

Remarks

A string specifying the reserve size. The default is a null reference (Nothing in Visual Basic).

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.

The reserve_size reserves file system space, which includes the metadata in the persistent store file, and not just the raw data. The metadata overhead and data compression must be considered when deriving the required file system space from the amount of database data.

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"
```

1.1.9.16 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;}
```

Remarks

A string specifying the location of the UltraLite engine executable. The default value is a null reference (Nothing in Visual Basic).

Only supply a StartLine (START) connection parameter if you are connecting to an engine that is not currently running.

Related Information

RuntimeType Property [page 249]

1.1.9.17 this[string keyword] Property

Specifies the value of the specified connection keyword.

Syntax

Visual Basic

```
Public Overrides Property Item (ByVal keyword As String) As Object
```

C#

```
public override object this[string keyword] {get;set;}
```

Remarks

An object representing the value of the specified connection keyword.

Connection keywords and the corresponding properties of the ULConnectionStringBuilder class are described in the table below:

Keyword	Corresponding Property
cache_size	ULConnectionStringBuilder.CacheSize
ce_file	ULConnectionStringBuilder.DatabaseOnDevice
con	ULConnectionStringBuilder.ConnectionName
dbkey	ULConnectionStringBuilder.DatabaseKey
dbn	ULConnectionStringBuilder.DatabaseName
nt_file	ULConnectionStringBuilder.DatabaseOnDesktop
pwd	ULConnectionStringBuilder.Password
reserve_size	ULConnectionStringBuilder.ReserveSize
start	ULConnectionStringBuilder.StartLine
uid	ULConnectionStringBuilder.UserID

Related Information

CacheSize Property [page 191]

DatabaseOnDevice Property [page 195]
ConnectionName Property [page 192]
DatabaseKey Property [page 193]
DatabaseName Property [page 193]
DatabaseOnDesktop Property [page 194]
Password Property [page 196]
ReserveSize Property [page 196]
StartLine Property [page 197]
UserID Property [page 199]

1.1.9.18 UserID Property

Specifies an authenticated user for the database.

Syntax

Visual Basic

```
Public Property UserID As String
```

C#

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

Remarks

A string specifying a database user ID. The default value is a null reference (Nothing in Visual Basic).

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 (with the ULConnection.GrantConnectTo method) and remove the DBA user (with the ULConnection.RevokeConnectFrom method).

Related Information

Password Property [page 196]
GrantConnectTo(string, string) Method [page 148]
RevokeConnectFrom(string) Method [page 152]

1.1.10 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
```

Members

All members of ULCreateParms, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULCreateParms() [page 203]	Initializes a ULCreateParms object with its default values.

Methods

Modifier and Type	Method	Description
public override string	ToString() [page 204]	Returns the string representation of this instance.

Properties

Modifier and Type	Property	Description
public bool	CaseSensitive [page 204]	Specifies whether the new database should be case sensitive when comparing string values.
public int	ChecksumLevel [page 205]	Specifies the level of database page checksums enabled for the new database.
public string	DateFormat [page 205]	Specifies the date format used for string conversions by the new database.
public ULDateOrder	DateOrder [page 206]	Specifies the date order used for string conversions by the new database.
public bool	FIPS [page 206]	Specifies whether the new database should be using AES_FIPS encryption or AES encryption.

Modifier and Type	Property	Description
public int	MaxHashSize [page 207]	Specifies the default maximum number of bytes to use for index hashing in the new database.
public int	NearestCentury [page 207]	Specifies the nearest century used for string conversions by the new database.
public bool	Obfuscate [page 208]	Specifies whether the new database should use obfuscation to encrypt the database.
public int	PageSize [page 208]	Specifies the page size, in bytes or kilobytes, of the new database.
public int	Precision [page 209]	Specifies the floating-point precision used for string conversions by the new database.
public int	Scale [page 210]	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.
public string	TimeFormat [page 210]	Specifies the time format used for string conversions by the new database.
public string	TimestampFormat [page 211]	Specifies the timestamp format used for string conversions by the new database.
public int	TimestampIncrement [page 211]	Specifies the minimum difference between two unique timestamps, in microseconds (1,000,000th of a second).
public bool	UTF8Encoding [page 212]	Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Remarks

A ULCreateParms object is used to specify the parameters for creating a database with the ULDatabaseManager.CreateDatabase method.

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 method. The resulting string can then be used as the createParms parameter of the ULDatabaseManager.CreateDatabase method.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows Mobile 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.DatabaseOnDevice = "\UltraLite\MyDatabase.udb"
ULConnection.DatabaseManager.CreateDatabase( _
   openParms.ToString(), _
   createParms.ToString() _
)
Dim conn As ULConnection = _
   New ULConnection( openParms.ToString() )
conn.Open()
```

The following code is the C# language equivalent:

```
// C#
ULCreateParms createParms = new ULCreateParms();
createParms.CaseSensitive = true;
createParms.UTF8Encoding = true;
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnDevice = ".udb";
ULConnection.DatabaseManager.CreateDatabase(
   openParms.ToString(),
   createParms.ToString()
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

In this section:

Specifies the nearest century used for string conversions by the new database.

Obfuscate Property [page 208]
Specifies whether the new database should use obfuscation to encrypt the database.

PageSize Property [page 208]
Specifies the page size, in bytes or kilobytes, of the new database.

Precision Property [page 209]
Specifies the floating-point precision used for string conversions by the new database.

Scale Property [page 210]
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 [page 210]
Specifies the time format used for string conversions by the new database.

TimestampFormat Property [page 211]
Specifies the timestamp format used for string conversions by the new database.

TimestampIncrement Property [page 211]
Specifies the minimum difference between two unique timestamps, in microseconds (1,000,000th of a second).

UTF8Encoding Property [page 212]
Specifies whether the new database should be using the UTF8 character set or the character set associated with the collation.

Related Information

GetDatabaseProperty(string) Method [page 251]
CreateDatabase(string, string) Method [page 241]
ToString() Method [page 204]

1.1.10.1 ULCreateParms() Constructor

Initializes a ULCreateParms object with its default values.

Syntax

Visual Basic

```
Public Sub ULCreateParms ()
```

C#

```
public ULCreateParms ()
```

1.1.10.2 ToString() Method

Returns the string representation of this instance.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

Returns

The string representation of this instance as a semicolon-separated list keyword=value pairs.

1.1.10.3 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;}
```

Remarks

True if the database should be case sensitive; false if the database should be case insensitive. The default is false.

This method 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.

1.1.10.4 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;}
```

Remarks

An integer specifying the checksum level. Valid values are 0, 1, and 2. The default is 2.

1.1.10.5 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;}
```

Remarks

A string specifying the date format. If the value is a null reference (Nothing in Visual Basic), the database uses "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).

1.1.10.6 DateOrder Property

Specifies the date order used for string conversions by the new database.

Syntax

Visual Basic

```
Public Property DateOrder As ULDateOrder
```

C#

```
public ULDateOrder DateOrder {get;set;}
```

Remarks

A ULDateOrder value identifying the date order for string conversions. The default is YMD.

Related Information

ULDateOrder Enumeration [page 568]

1.1.10.7 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;}
```

Remarks

True if the database should be encrypted using AES_FIPS, false if the database should be encrypted with AES. The default is false.

Encryption must be turned on by supplying a value for the EncryptionKey connection parameter when the new database is created. If FIPS is true and no encryption key is supplied, the ULDatabaseManager.CreateDatabase method fails with a missing encryption key error.

Related Information

EncryptionKey Property [page 181]
CreateDatabase(string, string) Method [page 241]

1.1.10.8 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;}
```

Remarks

An integer specifying the maximum hash size. The value must be in the range [0,32]. The default is 8.

1.1.10.9 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;}
```

Remarks

An integer specifying the nearest century. The value must be in the range [0,100]. The default is 50.

1.1.10.10 Obfuscate Property

Specifies whether the new database should use obfuscation to encrypt the database.

Syntax

Visual Basic

```
Public Property Obfuscate As Boolean
```

C#

```
public bool Obfuscate {get;set;}
```

Remarks

True if the database should be encrypted using obfuscation, false if the database should not be obfuscated. The default is false.

This option is ignored if FIPS encryption is turned on with the ULCreateParms.FIPS property. The encryption key is ignored if obfuscation is turned on and a value is supplied for the EncryptionKey connection parameter when the new database is created.

Related Information

FIPS Property [page 206]

1.1.10.11 PageSize Property

Specifies the page size, in bytes or kilobytes, of the new database.

Syntax

Visual Basic

```
Public Property PageSize As Integer
```

C#

```
public int PageSize {get;set;}
```

Remarks

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.

1.1.10.12 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;}
```

Remarks

An integer specifying the precision. The value must be in the range [1,127]. The default is 30.

Related Information

Scale Property [page 210]

1.1.10.13 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;}
```

Remarks

An integer specifying the scale. The value must be in the range [0,127]. The default is 6.

The Scale value must be less than or equal to the Precision value; otherwise, an error occurs when creating the database.

1.1.10.14 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;}
```

Remarks

A string specifying the time format. If the value is a null reference (Nothing in Visual Basic), the database uses "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).

1.1.10.15 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;}
```

Remarks

A string specifying the timestamp format. If the value is a null reference (Nothing in Visual Basic), the database uses "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).

1.1.10.16 TimestampIncrement Property

Specifies the minimum difference between two unique timestamps, in microseconds (1,000,000th of a second).

Syntax

Visual Basic

```
Public Property TimestampIncrement As Integer
```

C#

```
public int TimestampIncrement {get;set;}
```

Remarks

An integer specifying the timestamp increment. The value must be in the range [1,60000000]. The default is 1.

1.1.10.17 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;}
```

Remarks

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.

Choose to use the UTF8 character set 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.

1.1.11 ULCursorSchema Class

UL Ext: Represents the schema of an UltraLite.NET cursor.

Syntax

Visual Basic

```
Public MustInherit Class ULCursorSchema
```

C#

```
public abstract class ULCursorSchema
```

Members

All members of ULCursorSchema, including inherited members.

Methods

Modifier and Type	Method	Description
protected unsafe void	GetColumnCount() [page 215]	
public unsafe short	GetColumnID(string) [page 215]	Returns the column ID of the named column.
public string	GetColumnName(int) [page 216]	Returns the name of the column identified by the specified column ID.
public unsafe int	GetColumnPrecision(int) [page 217]	Returns the precision of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).
public unsafe int	GetColumnScale(int) [page 218]	Returns the scale of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).
public unsafe int	GetColumnSize(int) [page 219]	Returns the size of the column identified by the specified column ID if the column is a sized column (the BINARY or CHAR SQL types).
public string	GetColumnSQLName(int) [page 220]	Returns the name of the column identified by the specified column ID.
public unsafe ULDbType	GetColumnULDbType(int) [page 221]	Returns the UltraLite.NET data type of the column identified by the specified column ID.
public unsafe DataTable	GetSchemaTable() [page 222]	Returns a System.Data.DataTable that describes the column schema of the ULDataReader object.
protected virtual void	VerifyOpen() [page 222]	

Properties

Modifier and Type	Property	Description
public short	ColumnCount [page 223]	Returns the number of columns in the cursor.
public bool	IsOpen [page 223]	Checks whether the cursor schema is currently open.
public abstract string	Name [page 224]	Returns the name of the cursor.

Remarks

This class is an abstract base class of the ULTableSchema class and the ULResultSetSchema class.

> **i** Note
>
> For users porting from the Sap.UltraLite namespace, Column IDs are 0-based, not 1-based as they are in the Sap.UltraLite namespace.

In this section:

Related Information

ULTableSchema Class [page 537]

ULResultSetSchema Class [page 453]

1.1.11.1 GetColumnCount() Method

Syntax

Visual Basic

```
Protected Sub GetColumnCount ()
```

C#

```
protected unsafe void GetColumnCount ()
```

1.1.11.2 GetColumnID(string) Method

Returns the column ID of the named column.

Syntax

Visual Basic

```
Public Function GetColumnID (ByVal name As String) As Short
```

C#

```
public unsafe short GetColumnID (string name)
```

Parameters

name The name of the column.

Returns

The column ID of the named column.

Exceptions

ULException class A SQL error occurred.

Remarks

Column IDs range from 0 to ColumnCount-1, inclusive.

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.

Related Information

ColumnCount Property [page 223]
ColumnCount Property [page 223]

1.1.11.3 GetColumnName(int) 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 The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

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.

Exceptions

ULException class A SQL error occurred.

Remarks

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.

Related Information

ColumnCount Property [page 223]
ColumnCount Property [page 223]

1.1.11.4 GetColumnPrecision(int) Method

Returns the precision of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).

Syntax

Visual Basic

```
Public Function GetColumnPrecision (ByVal columnID As Integer) As Integer
```

C#

```
public unsafe int GetColumnPrecision (int columnID)
```

Parameters

columnID The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

The precision of the specified numeric column.

Exceptions

ULException class A SQL error occurred.

Related Information

GetColumnULDbType(int) Method [page 221]
ColumnCount Property [page 223]

1.1.11.5 GetColumnScale(int) Method

Returns the scale of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).

Syntax

Visual Basic

```
Public Function GetColumnScale (ByVal columnID As Integer) As Integer
```

C#

```
public unsafe int GetColumnScale (int columnID)
```

Parameters

columnID The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

The scale of the specified numeric column.

Exceptions

ULException class A SQL error occurred.

Related Information

GetColumnULDbType(int) Method [page 221]
ColumnCount Property [page 223]

1.1.11.6 GetColumnSize(int) Method

Returns the size of the column identified by the specified column ID if the column is a sized column (the BINARY or CHAR SQL types).

Syntax

Visual Basic

```
Public Function GetColumnSize (ByVal columnID As Integer) As Integer
```

C#

```
public unsafe int GetColumnSize (int columnID)
```

Parameters

columnID The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

The size of the specified sized column.

Exceptions

ULException class A SQL error occurred.

Related Information

GetColumnULDbType(int) Method [page 221]
ColumnCount Property [page 223]

1.1.11.7 GetColumnSQLName(int) 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 The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

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.

Exceptions

ULException class A SQL error occurred.

Remarks

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 method because the GetColumnSQLName method always returns just the name of the column (without the table name as a prefix)

for non-aliased, non-computed columns. 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.

Related Information

ColumnCount Property [page 223]
GetColumnName(int) Method [page 216]
ColumnCount Property [page 223]

1.1.11.8 GetColumnULDbType(int) 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 unsafe ULDbType GetColumnULDbType (int columnID)
```

Parameters

columnID The ID of the column. The value must be in the range [0,ColumnCount-1].

Returns

A ULDbType enumerated integer.

Exceptions

ULException class A SQL error occurred.

Related Information

ColumnCount Property [page 223]
ColumnCount Property [page 223]
ULDbType Enumeration [page 568]

1.1.11.9 GetSchemaTable() Method

Returns a System.Data.DataTable that describes the column schema of the ULDataReader object.

Syntax

Visual Basic

```
Public Function GetSchemaTable () As DataTable
```

C#

```
public unsafe DataTable GetSchemaTable ()
```

Returns

A System.Data.DataTable that describes the column schema.

Related Information

GetSchemaTable() Method [page 288]
ULDataReader Class [page 260]

1.1.11.10 VerifyOpen() Method

Syntax

Visual Basic

```
Protected Overridable Sub VerifyOpen ()
```

C#

```
protected virtual void VerifyOpen ()
```

1.1.11.11 ColumnCount Property

Returns the number of columns in the cursor.

Syntax

Visual Basic

```
Public ReadOnly Property ColumnCount As Short
```

C#

```
public short ColumnCount {get;}
```

Remarks

The number of columns in the cursor or 0 if the cursor schema is closed.

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.

1.1.11.12 IsOpen Property

Checks whether the cursor schema is currently open.

Syntax

Visual Basic

```
Public ReadOnly Property IsOpen As Boolean
```

C#

```
public bool IsOpen {get;}
```

Remarks

True if the cursor schema is currently open; false if the cursor schema is closed.

1.1.11.13 Name Property

Returns the name of the cursor.

Syntax

Visual Basic

```
Public ReadOnly Property Name As String
```

C#

```
public abstract string Name {get;}
```

Remarks

The name of the cursor as a string.

1.1.12 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

```
Public NotInheritable Class ULDataAdapter Inherits
System.Data.Common.DbDataAdapter Implements System.ICloneable
```

C#

```
public sealed class ULDataAdapter : System.Data.Common.DbDataAdapter,
System.ICloneable
```

Members

All members of ULDataAdapter, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULDataAdapter [page 227]	Initializes a ULDataAdapter object.

Methods

Modifier and Type	Method	Description
protected override void	Dispose(bool) [page 230]	Releases the unmanaged resources used by the ULDataAdapter object and optionally releases the managed resources.
protected override int	Fill [page 231]	See individual topics
protected override DataTable[]	FillSchema [page 232]	See individual topics
public new ULParameter[]	GetFillParameters() [page 234]	Returns the parameters set by the user when executing a SELECT statement.
protected override int	Update(DataRow[], DataTableMapping) [page 234]	

Properties

Modifier and Type	Property	Description
public new ULCommand	DeleteCommand [page 235]	Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.
public new ULCommand	InsertCommand [page 235]	Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.
public new ULCommand	SelectCommand [page 236]	Specifies a ULCommand that is used during the System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet) or System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataSet,System.Data.SchemaType) method calls to obtain a result set from the database for copying into a System.Data.DataSet.
public new DataTableMappingCollection	TableMappings [page 237]	Returns a collection that provides the master mapping between a source table and a System.Data.DataTable
public new ULCommand	UpdateCommand [page 238]	Specifies a ULCommand object that is executed against the database when the System.Data.Common.DbDataAdapter.Update method is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Events

Modifier and Type	Event	Description
public ULRowUpdatedEventHandler	RowUpdated [page 238]	Occurs during an update after a command is executed against the data source.
public ULRowUpdatingEventHandler	RowUpdating [page 239]	Occurs during an update before a command is executed against the data source.

Remarks

The System.Data.DataSet provides a way to work with data offline; that is, away from your UltraLite database. The ULDataAdapter class 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 ULTable classes provide more efficient access to data.

In this section:

ULDataAdapter Constructor [page 227]
Initializes a ULDataAdapter object.

Dispose(bool) Method [page 230]
Releases the unmanaged resources used by the ULDataAdapter object and optionally releases the managed resources.

Fill Method [page 231]

FillSchema Method [page 232]

GetFillParameters() Method [page 234]
Returns the parameters set by the user when executing a SELECT statement.

Update(DataRow[], DataTableMapping) Method [page 234]

DeleteCommand Property [page 235]
Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

InsertCommand Property [page 235]
Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

SelectCommand Property [page 236]
Specifies a ULCommand that is used during the System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet) or System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataSet,System.Data.SchemaType) method calls to obtain a result set from the database for copying into a System.Data.DataSet.

TableMappings Property [page 237]
Returns a collection that provides the master mapping between a source table and a System.Data.DataTable

UpdateCommand Property [page 238]
Specifies a ULCommand object that is executed against the database when the System.Data.Common.DbDataAdapter.Update method is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

RowUpdated Event [page 238]
Occurs during an update after a command is executed against the data source.

RowUpdating Event [page 239]
Occurs during an update before a command is executed against the data source.

Related Information

ULDataReader Class [page 260]
ULTable Class [page 507]

1.1.12.1 ULDataAdapter Constructor

Initializes a ULDataAdapter object.

Overload List

Modifier and Type	Overload name	Description
public	ULDataAdapter() [page 228]	Initializes a ULDataAdapter object.
public	ULDataAdapter(ULCommand) [page 228]	Initializes a ULDataAdapter object with the specified SELECT statement.
public	ULDataAdapter(string, ULConnection) [page 229]	Initializes a ULDataAdapter object with the specified SELECT statement and connection.
public	ULDataAdapter(string, string) [page 230]	Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

In this section:

ULDataAdapter() Constructor [page 228]
Initializes a ULDataAdapter object.

ULDataAdapter(ULCommand) Constructor [page 228]
Initializes a ULDataAdapter object with the specified SELECT statement.

ULDataAdapter(string, ULConnection) Constructor [page 229]
Initializes a ULDataAdapter object with the specified SELECT statement and connection.

ULDataAdapter(string, string) Constructor [page 230]
Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

1.1.12.1.1 ULDataAdapter() Constructor

Initializes a ULDataAdapter object.

Syntax

Visual Basic

```
Public Sub ULDataAdapter ()
```

C#

```
public ULDataAdapter ()
```

Related Information

ULDataAdapter(ULCommand) Constructor [page 228]
ULDataAdapter(string, ULConnection) Constructor [page 229]
ULDataAdapter(string, string) Constructor [page 230]

1.1.12.1.2 ULDataAdapter(ULCommand) Constructor

Initializes a ULDataAdapter object with the specified SELECT statement.

Syntax

Visual Basic

```
Public Sub ULDataAdapter (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.

Related Information

ULDataAdapter() Constructor [page 228]
ULDataAdapter(string, ULConnection) Constructor [page 229]
ULDataAdapter(string, string) Constructor [page 230]
ULCommand Class [page 45]

1.1.12.1.3 ULDataAdapter(string, ULConnection) Constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection.

Syntax

Visual Basic

```
Public Sub ULDataAdapter (
    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 method of the ULDataAdapter object.

selectConnection A ULConnection object that defines a connection to a database.

Related Information

ULDataAdapter() Constructor [page 228]
ULDataAdapter(ULCommand) Constructor [page 228]
ULDataAdapter(string, string) Constructor [page 230]
SelectCommand Property [page 236]
ULConnection Class [page 108]

1.1.12.1.4 ULDataAdapter(string, string) Constructor

Initializes a ULDataAdapter object with the specified SELECT statement and connection string.

Syntax

Visual Basic

```
Public Sub ULDataAdapter (
    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 method.
selectConnectionString A connection string for an UltraLite.NET database.

Related Information

ULDataAdapter() Constructor [page 228]
ULDataAdapter(ULCommand) Constructor [page 228]
ULDataAdapter(string, ULConnection) Constructor [page 229]
SelectCommand Property [page 236]

1.1.12.2 Dispose(bool) Method

Releases the unmanaged resources used by the ULDataAdapter object and optionally releases the managed resources.

Syntax

Visual Basic

```
Protected Overrides Sub Dispose (ByVal disposing As Boolean)
```

C#

```
protected override void Dispose (bool disposing)
```

Parameters

disposing When true, disposes of both managed and unmanaged resources. When false, disposes of only the unmanaged resources.

1.1.12.3 Fill Method

Overload List

Modifier and Type	Overload name	Description
protected override int	Fill(DataSet, int, int, string, IDbCommand, CommandBehavior) [page 231]	
protected override int	Fill(DataTable[], int, int, IDbCommand, CommandBehavior) [page 232]	

In this section:

1.1.12.3.1 Fill(DataSet, int, int, string, IDbCommand, CommandBehavior) Method

Syntax

Visual Basic

```
Protected Overrides Function Fill (
    ByVal dataSet As DataSet,
    ByVal startRecord As Integer,
    ByVal maxRecords As Integer,
    ByVal srcTable As String,
    ByVal cmd As IDbCommand,
    ByVal behavior As CommandBehavior
) As Integer
```

C#

```
protected override int Fill (
    DataSet dataSet,
    int startRecord,
    int maxRecords,
    string srcTable,
```

```
    IDbCommand cmd,
    CommandBehavior behavior
)
```

1.1.12.3.2 Fill(DataTable[], int, int, IDbCommand, CommandBehavior) Method

Syntax

Visual Basic

```
Protected Overrides Function Fill (
    ByVal dataTables As DataTable(),
    ByVal startRecord As Integer,
    ByVal maxRecords As Integer,
    ByVal cmd As IDbCommand,
    ByVal behavior As CommandBehavior
) As Integer
```

C#

```
protected override int Fill (
    DataTable[] dataTables,
    int startRecord,
    int maxRecords,
    IDbCommand cmd,
    CommandBehavior behavior
)
```

1.1.12.4 FillSchema Method

Overload List

Modifier and Type	Overload name	Description
protected override DataTable[]	FillSchema(DataSet, SchemaType, IDbCommand, string, CommandBehavior) [page 233]	
protected override DataTable	FillSchema(DataTable, SchemaType, IDbCommand, CommandBehavior) [page 233]	

In this section:

FillSchema(DataTable, SchemaType, IDbCommand, CommandBehavior) Method [page 233]

1.1.12.4.1 FillSchema(DataSet, SchemaType, IDbCommand, string, CommandBehavior) Method

Syntax

Visual Basic

```
Protected Overrides Function FillSchema (
    ByVal dataSet As DataSet,
    ByVal schemaType As SchemaType,
    ByVal cmd As IDbCommand,
    ByVal srcTable As String,
    ByVal behavior As CommandBehavior
) As DataTable()
```

C#

```
protected override DataTable[] FillSchema (
    DataSet dataSet,
    SchemaType schemaType,
    IDbCommand cmd,
    string srcTable,
    CommandBehavior behavior
)
```

1.1.12.4.2 FillSchema(DataTable, SchemaType, IDbCommand, CommandBehavior) Method

Syntax

Visual Basic

```
Protected Overrides Function FillSchema (
    ByVal dataTable As DataTable,
    ByVal schemaType As SchemaType,
    ByVal cmd As IDbCommand,
    ByVal behavior As CommandBehavior
) As DataTable
```

C#

```
protected override DataTable FillSchema (
    DataTable dataTable,
    SchemaType schemaType,
    IDbCommand cmd,
    CommandBehavior behavior
)
```

1.1.12.5 GetFillParameters() Method

Returns the parameters set by the user when executing a SELECT statement.

Syntax

Visual Basic

```
Public Shadows Function GetFillParameters () As ULParameter()
```

C#

```
public new ULParameter[] GetFillParameters ()
```

Returns

An array of ULParameter objects that contains the parameters set by the user.

Remarks

This is the strongly-typed version of the System.Data.Common.DbDataAdapter.GetFillParameters method.

Related Information

ULParameter Class [page 372]

1.1.12.6 Update(DataRow[], DataTableMapping) Method

Syntax

Visual Basic

```
Protected Overrides Function Update (
    ByVal dataRows As DataRow(),
    ByVal tm As DataTableMapping
) As Integer
```

C#

```
protected override int Update (
    DataRow[] dataRows,
    DataTableMapping tm
```

```
)
```

1.1.12.7 DeleteCommand Property

Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

Syntax

Visual Basic

```
Public Shadows Property DeleteCommand As ULCommand
```

C#

```
public new ULCommand DeleteCommand {get;set;}
```

Remarks

A ULCommand object that is executed to delete rows in the database that correspond to deleted rows in the System.Data.DataSet.

When the DeleteCommand property is assigned to an existing ULCommand object, the ULCommand object is not cloned. The DeleteCommand property maintains a reference to the existing ULCommand object.

This is the strongly-typed version of the System.Data.IDbDataAdapter.DeleteCommand and System.Data.Common.DbDataAdapter.DeleteCommand properties.

Related Information

ULCommand Class [page 45]

1.1.12.8 InsertCommand Property

Specifies a ULCommand object that is executed against the database when the DbDataAdapter.Update method is called to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

Syntax

Visual Basic

```
Public Shadows Property InsertCommand As ULCommand
```

C#

```
public new ULCommand InsertCommand {get;set;}
```

Remarks

A ULCommand object that is executed to insert rows in the database that correspond to inserted rows in the System.Data.DataSet.

When the InsertCommand property is assigned to an existing ULCommand object, the ULCommand object is not cloned. The InsertCommand property maintains a reference to the existing ULCommand object.

This is the strongly-typed version of System.Data.IDbDataAdapter.InsertCommand and System.Data.Common.DbDataAdapter.InsertCommand properties.

Related Information

ULCommand Class [page 45]

1.1.12.9 SelectCommand Property

Specifies a ULCommand that is used during the System.Data.Common.DbDataAdapter.Fill(System.Data.DataSet) or System.Data.Common.DbDataAdapter.FillSchema(System.Data.DataSet,System.Data.SchemaType) method calls to obtain a result set from the database for copying into a System.Data.DataSet.

Syntax

Visual Basic

```
Public Shadows Property SelectCommand As ULCommand
```

C#

```
public new ULCommand SelectCommand {get;set;}
```

Remarks

A ULCommand object that is executed to fill the System.Data.DataSet.

When SelectCommand property is assigned to an existing ULCommand object, the ULCommand object is not cloned. The SelectCommand property maintains a reference to the existing ULCommand object.

If the SelectCommand property does not return any rows, then 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 the System.Data.IDbDataAdapter.SelectCommand and System.Data.Common.DbDataAdapter.SelectCommand properties.

Related Information

ULCommand Class [page 45]
ULDataAdapter(ULCommand) Constructor [page 228]
ULDataAdapter Class [page 224]
ULDataAdapter Class [page 224]

1.1.12.10 TableMappings Property

Returns a collection that provides the master mapping between a source table and a System.Data.DataTable

Syntax

Visual Basic

```
Public ReadOnly Shadows Property TableMappings As
DataTableMappingCollection
```

C#

```
public new DataTableMappingCollection TableMappings {get;}
```

Remarks

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.

When reconciling changes, the ULDataAdapter object 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 object.

This is the strongly-typed version of the System.Data.IDataAdapter.TableMappings property.

1.1.12.11 UpdateCommand Property

Specifies a ULCommand object that is executed against the database when the System.Data.Common.DbDataAdapter.Update method is called to update rows in the database that correspond to updated rows in the System.Data.DataSet.

Syntax

Visual Basic

```
Public Shadows Property UpdateCommand As ULCommand
```

C#

```
public new ULCommand UpdateCommand {get;set;}
```

Remarks

A ULCommand object that is executed to update rows in the database that correspond to updated rows in the System.Data.DataSet.

When UpdateCommand is assigned to an existing ULCommand object, the ULCommand object is not cloned. The UpdateCommand property maintains a reference to the existing ULCommand object.

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 property of the ULCommand object.

This is the strongly-typed version of the System.Data.IDbDataAdapter.UpdateCommand and System.Data.Common.DbDataAdapter.DeleteCommand properties.

Related Information

ULCommand Class [page 45]
UpdatedRowSource Property [page 92]

1.1.12.12 RowUpdated Event

Occurs during an update after a command is executed against the data source.

Syntax

Visual Basic

```
Public Event RowUpdated  As ULRowUpdatedEventHandler
```

C#

```
public ULRowUpdatedEventHandler RowUpdated;
```

Remarks

When an attempt to update is made, the event fires.

To process row updated events, you must create a ULRowUpdatedEventHandler delegate and attach it to this event.

Related Information

ULRowUpdatedEventHandler(object, ULRowUpdatedEventArgs) Delegate [page 563]

1.1.12.13 RowUpdating Event

Occurs during an update before a command is executed against the data source.

Syntax

Visual Basic

```
Public Event RowUpdating  As ULRowUpdatingEventHandler
```

C#

```
public ULRowUpdatingEventHandler RowUpdating;
```

Remarks

When an attempt to update is made, the event fires.

To process row updating events, you must create a ULRowUpdatingEventHandler delegate and attach it to this event.

Related Information

ULRowUpdatedEventHandler(object, ULRowUpdatedEventArgs) Delegate [page 563]

1.1.13 ULDatabaseManager Class

UL Ext: Provides static methods for creating, deleting, and validating databases.

Syntax

Visual Basic

```
Public NotInheritable Class ULDatabaseManager
```

C#

```
public sealed class ULDatabaseManager
```

Members

All members of ULDatabaseManager, including inherited members.

Methods

Modifier and Type	Method	Description
public static void	CreateDatabase(string, string) [page 241]	Creates a new UltraLite database.
public static void	DropDatabase(string) [page 243]	Deletes the specified database.
public static void	SetActiveSyncListener(string, ULActiveSyncListener) [page 244]	Specifies the listener object used to process ActiveSync calls from the MobiLink provider for ActiveSync.
public static void	SetServerSyncListener(string, string, ULServerSyncListener) [page 246]	Specifies the listener object used to process the specified server synchronization message.
public static void	SignalSyncIsComplete() [page 247]	Signals the MobiLink provider for ActiveSync that an application has completed synchronization.
public static void	ValidateDatabase(string, ULDBValid) [page 247]	Performs low level and index validation on a database.

Properties

Modifier and Type	Property	Description
public ULRuntimeType	RuntimeType [page 249]	Specifies the UltraLite.NET runtime type.

Remarks

To use the UltraLite Engine runtime of UltraLite.NET, set the ULDatabaseManager.RuntimeType property 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
```

The following code is the C# language equivalent:

```
// C#
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT;
ULConnection conn = new ULConnection();
// The RuntimeType is now locked
```

In this section:

1.1.13.1 CreateDatabase(string, string) Method

Creates a new UltraLite database.

Syntax

Visual Basic

```
Public Shared Sub CreateDatabase (
    ByVal connString As String,
    ByVal createParms As String
)
```

C#

```
public static void CreateDatabase (
```

```
        string connString,
        string createParms
    )
```

Parameters

connString The parameters for identifying a database in the form of a semicolon-separated list of keyword=value pairs.

createParms The parameters used to configure the new database in the form of a semicolon-separated list of keyword=value pairs.

Exceptions

ULException class A SQL error occurred.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows Mobile device then opens a connection to it.

```
' Visual Basic
Dim openParms As ULConnectionParms = New ULConnectionParms
openParms.DatabaseOnDevice = ".udb"
ULDatabaseManager.CreateDatabase( _
   openParms.ToString(), _
   "" _
)
Dim conn As ULConnection = _
   New ULConnection( openParms.ToString() )
conn.Open()
```

The following code is the C# language equivalent:

```
// C#
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnDevice = ".udb";
ULDatabaseManager.CreateDatabase(
   openParms.ToString(),
   ""
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

Related Information

Open() Method [page 149]
ULConnectionParms Class [page 173]
ULCreateParms Class [page 200]

1.1.13.2 DropDatabase(string) Method

Deletes the specified database.

Syntax

Visual Basic

```
Public Shared Sub DropDatabase (ByVal connString As String)
```

C#

```
public static void DropDatabase (string connString)
```

Parameters

connString The parameters for identifying a database in the form of a semicolon-separated list of keyword=value pairs.

Exceptions

ULException class A SQL error occurred.

Remarks

You cannot drop a database that has open connections.

Example

The following code creates the database \UltraLite\MyDatabase.udb on a Windows Mobile device then opens a connection to it:

```
' Visual Basic
Dim connParms As ULConnectionParms = New ULConnectionParms
connParms.DatabaseOnDevice = ".udb"
ULConnection.DatabaseManager.DropDatabase( _
   connParms.ToString() _
)
```

The following code is the C# language equivalent:

```
// C#
ULConnectionParms connParms = new ULConnectionParms();
connParms.DatabaseOnDevice = ".udb";
ULConnection.DatabaseManager.DropDatabase(
   connParms.ToString()
);
ULConnection conn = new ULConnection( openParms.ToString() );
conn.Open();
```

Related Information

Open() Method [page 149]
ULConnectionParms Class [page 173]

1.1.13.3 SetActiveSyncListener(string, ULActiveSyncListener) Method

Specifies the listener object used to process Microsoft ActiveSync calls from the MobiLink provider for Microsoft ActiveSync.

Syntax

Microsoft Visual Basic

```
Public Shared Sub SetActiveSyncListener (
    ByVal appClassName As String,
    ByVal listener As ULActiveSyncListener
)
```

C#

```
public static 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 Microsoft ActiveSync.

listener The ULActiveSyncListener object. Use null (Nothing in Microsoft Visual Basic) to remove the previous listener.

Exceptions

ULException class A SQL error occurred.

Remarks

The *appClassName* parameter is the unique identifier used to identify the application. The application can only use one *appClassName* value at a time. While a listener is registered with a particular *appClassName* value, calls to the SetServerSyncListener or SetActiveSyncListener methods with a different *appClassName* value fail.

To remove the Microsoft ActiveSync listener, call the SetActiveSyncListener method with a null reference (Nothing in Microsoft Visual Basic) as the *listener* parameter.

To remove all listeners, call the SetServerSyncListener method with a null reference (Nothing in Microsoft Visual Basic) for all parameters.

Applications should remove all listeners prior to exiting.

Related Information

SetServerSyncListener(string, string, ULServerSyncListener) Method [page 246]
ULActiveSyncListener Interface [page 8]
ActiveSyncInvoked(bool) Method [page 8]

1.1.13.4 SetServerSyncListener(string, string, ULServerSyncListener) Method

Specifies the listener object used to process the specified server synchronization message.

Syntax

Visual Basic

```
Public Shared Sub SetServerSyncListener (
    ByVal messageName As String,
    ByVal appClassName As String,
    ByVal listener As ULServerSyncListener
)
```

C#

```
public static 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The *appClassName* parameter is the unique identifier used to identify the application. The application may only use one *appClassName* value at a time. While a listener is registered with a particular *appClassName* value, calls to the SetServerSyncListener or SetActiveSyncListener methods with a different *appClassName* value fail.

To remove the listener for a particular message, call the SetServerSyncListener method with a null reference (Nothing in Visual Basic) as the *listener* parameter.

To remove all listeners, call the SetServerSyncListener method with a null reference (Nothing in Visual Basic) for all parameters.

Applications should remove all listeners before exiting.

Related Information

SetActiveSyncListener(string, ULActiveSyncListener) Method [page 244]
ServerSyncInvoked(string) Method [page 464]

1.1.13.5 SignalSyncIsComplete() Method

Signals the MobiLink provider for ActiveSync that an application has completed synchronization.

Syntax

Visual Basic

```
Public Shared Sub SignalSyncIsComplete ()
```

C#

```
public static void SignalSyncIsComplete ()
```

Related Information

ActiveSyncInvoked(bool) Method [page 8]

1.1.13.6 ValidateDatabase(string, ULDBValid) Method

Performs low level and index validation on a database.

Syntax

Visual Basic

```
Public Shared Sub ValidateDatabase (
    ByVal start_parms As String,
    ByVal how As ULDBValid
)
```

C#

```
public static void ValidateDatabase (
```

```
        string start_parms,
        ULDBValid how
    )
```

Parameters

start_parms The parameters for identifying a database in the form of a semicolon-separated list of keyword=value pairs.

how Describes how to validate the database.

Exceptions

ULException class A SQL error occurred.

Example

The following code validates indexes for the database \UltraLite\MyDatabase.udb under Windows Mobile:

```
' Visual Basic
Dim openParms As ULConnectionParms = New ULConnectionParms
openParms.DatabaseOnDevice = "\UltraLite\MyDatabase.udb"
ULConnection.DatabaseManager.ValidateDatabase( _
   openParms.ToString(), Sap.Data.UltraLite.ULVF_INDEX )
```

The following code is the C# language equivalent:

```
// C#
ULConnectionParms openParms = new ULConnectionParms();
openParms.DatabaseOnDevice = ".udb";
ULConnection.DatabaseManager.ValidateDatabase(
   openParms.ToString(), Sap.Data.UltraLite.ULVF_INDEX );
```

Related Information

ValidateDatabase(ULDBValid) Method [page 161]
ValidateDatabase(ULDBValid, string) Method [page 162]
ULDBValid Enumeration [page 571]
ULConnectionParms Class [page 173]

1.1.13.7 RuntimeType Property

Specifies the UltraLite.NET runtime type.

Syntax

Visual Basic

```
Public Shared Property RuntimeType As ULRuntimeType
```

C#

```
public ULRuntimeType RuntimeType {get;set;}
```

Remarks

The runtime type must be selected before using any other UltraLite.NET API.

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
```

The following code is the C# language equivalent:

```
// C#
ULDatabaseManager.RuntimeType = ULRuntimeType.UL_ENGINE_CLIENT;
ULConnection conn = new ULConnection();
// The RuntimeType is now locked
```

Related Information

ULRuntimeType Enumeration [page 572]

1.1.14 ULDatabaseSchema Class

UL Ext: Represents the schema of an UltraLite.NET database.

Syntax

Visual Basic

```
Public NotInheritable Class ULDatabaseSchema
```

C#

```
public sealed class ULDatabaseSchema
```

Members

All members of ULDatabaseSchema, including inherited members.

Methods

Modifier and Type	Method	Description
public string	GetDatabaseProperty(string) [page 251]	Returns the value of the specified database property.
public string	GetPublicationName(int) [page 253]	Returns the name of the publication identified by the specified publication ID.
public string	GetTableName(int) [page 255]	Returns the name of the table identified by the specified table ID.
public void	SetDatabaseOption(string, string) [page 256]	Sets the value for the specified database option.

Properties

Modifier and Type	Property	Description
public unsafe bool	IsCaseSensitive [page 257]	Checks whether the database is case sensitive.
public bool	IsOpen [page 258]	Determines whether the database schema is open.
public unsafe int	PublicationCount [page 258]	Counts the number of publications in the database.
public unsafe int	TableCount [page 259]	Counts the number of tables in the database.

Remarks

There is no constructor for this class. A ULDatabaseSchema object is attached to a connection as its ULConnection.Schema object and is only valid while that connection is open.

In this section:

Related Information

Schema Property [page 168]

1.1.14.1 GetDatabaseProperty(string) 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.

Returns

The value of the property as a string.

Exceptions

ULException class A SQL error occurred.

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 values 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.
Collation	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.
File	The file name of the database.
global_database_id	The value of the global_database_id option used for global autoincrement columns.

Property	Description
isolation_level	The value of the isolation_level option used for controlling the degree to which the operations in one transaction are visible to the operations in other concurrent transactions. This value is set on a per connection basis.
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 value 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 microseconds (1,000,000th of a second).

Related Information

SetDatabaseOption(string, string) Method [page 256]
UserID Property [page 182]
Password Property [page 181]

1.1.14.2 GetPublicationName(int) Method

Returns the name of the publication identified by the specified publication ID.

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].

Returns

The publication name as a string.

Exceptions

ULException class A SQL error occurred.

Remarks

> **i** Note
>
> Publication IDs and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs and counts after a schema upgrade.

Related Information

PublicationCount Property [page 258]
PublicationCount Property [page 258]

1.1.14.3 GetTableName(int) 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].

Returns

The table name as a string.

Exceptions

ULException class A SQL error occurred.

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.

Related Information

TableCount Property [page 259]

1.1.14.4 SetDatabaseOption(string, string) 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.

Exceptions

ULException class A SQL error occurred.

Remarks

Setting a database option results in a commit being performed.

Using this method while a transaction is active may cause unpredictable results and is not recommended. The call changes the isolation level of the connection, but does not update the ULTransaction.IsolationLevel value.

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 the ULConnection.INVALID_DATABASE_ID value (used to indicate that the database ID has not been set for the current database).

Option	Description
isolation_level	The value used to control the degree to which the operations in one transaction are visible to the operations in other concurrent transactions. The value must be one of "read_uncommitted" or "read_committed". The default is "read_committed". Setting the isolation_level on a connection to "read_uncommitted" is equivalent to wrapping all operations on that connection with BeginTransaction(System.Data.IsolationLevel.ReadUncommitted) and Commit() calls. Similarly, "read_committed" is equivalent to System.Data.IsolationLevel.ReadCommitted. SetDatabaseOption() should not be used to set the current transaction's isolation level; use BeginTransaction(IsolationLevel) instead. UltraLite's definition of each isolation level is slightly different than ADO.NET's documentation of IsolationLevel. This value is set on a per connection basis.
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.

Related Information

GetDatabaseProperty(string) Method [page 251]

1.1.14.5 IsCaseSensitive Property

Checks whether the database is case sensitive.

Syntax

Visual Basic

```
Public ReadOnly Property IsCaseSensitive As Boolean
```

C#

```
public unsafe bool IsCaseSensitive {get;}
```

Remarks

True if the database is case sensitive, and false if the database is case insensitive.

Database case sensitivity affects how indexes on tables and result sets are sorted. Case sensitivity also affects how the ULConnectionParms.UserID and ULConnectionParms.Password values are verified.

Related Information

GetDatabaseProperty(string) Method [page 251]
UserID Property [page 182]
Password Property [page 181]

1.1.14.6 IsOpen Property

Determines whether the database schema is open.

Syntax

Visual Basic

```
Public ReadOnly Property IsOpen As Boolean
```

C#

```
public bool IsOpen {get;}
```

Remarks

True if this database schema is currently open, false if this database schema is currently closed.

A ULDatabaseSchema object is open only if the connection it is attached to is open.

1.1.14.7 PublicationCount Property

Counts the number of publications in the database.

Syntax

Visual Basic

```
Public ReadOnly Property PublicationCount As Integer
```

C#

```
public unsafe int PublicationCount {get;}
```

Remarks

The number of publications in the database.

Publication IDs range from 1 to the PublicationCount value, inclusively.

> **i** Note
>
> Publication IDs and counts may change during a schema upgrade. To correctly identify a publication, access it by name or refresh the cached IDs and counts after a schema upgrade.

Related Information

GetPublicationName(int) Method [page 253]

1.1.14.8 TableCount Property

Counts the number of tables in the database.

> **‹≡› Syntax**
>
> **Visual Basic**
>
> ```
> Public ReadOnly Property TableCount As Integer
> ```
>
> **C#**
>
> ```
> public unsafe int TableCount {get;}
> ```

Remarks

The number of tables in the database.

Table IDs range from 1 to the TableCount value, inclusively.

> **i** 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.

1.1.15 ULDataReader Class

Represents a read-only bi-directional cursor in an UltraLite database.

Syntax

Visual Basic

```
Public Class ULDataReader Inherits System.Data.Common.DbDataReader
Implements System.ComponentModel.IListSource
```

C#

```
public class ULDataReader : System.Data.Common.DbDataReader,
System.ComponentModel.IListSource
```

Members

All members of ULDataReader, including inherited members.

Methods

Modifier and Type	Method	Description
public override void	Close() [page 266]	Closes the cursor.
protected override void	Dispose(bool) [page 267]	
public override unsafe bool	GetBoolean(int) [page 267]	Returns the value for the specified column as a System.Boolean.
public override unsafe byte	GetByte(int) [page 268]	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
public unsafe byte[]	GetBytes [page 269]	UL Ext: Returns the value for the specified column as an array of System.Bytes values.
public override char	GetChar(int) [page 272]	This method is not supported in UltraLite.NET.
public override unsafe long	GetChars(int, long, char[], int, int) [page 273]	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.
public override string	GetDataTypeName(int) [page 274]	Returns the name of the specified column's provider data type.
public override unsafe DateTime	GetDateTime(int) [page 275]	Returns the value for the specified column as a System.DateTime type with millisecond accuracy.
protected override DbDataReader	GetDbDataReader(int) [page 276]	

Modifier and Type	Method	Description
public override decimal	GetDecimal(int) [page 276]	Returns the value for the specified column as a System.Decimal type.
public override unsafe double	GetDouble(int) [page 277]	Returns the value for the specified column as a System.Double type.
public override IEnumerator	GetEnumerator() [page 278]	Returns an System.Collections.IEnumerator value that iterates through the ULDataReader object.
public override Type	GetFieldType(int) [page 279]	Returns the System.Type value most appropriate for the specified column.
public override unsafe float	GetFloat(int) [page 280]	Returns the value for the specified column as a System.Single type.
public override unsafe Guid	GetGuid(int) [page 281]	Returns the value for the specified column as a UUID (System.Guid) type.
public override unsafe short	GetInt16(int) [page 282]	Returns the value for the specified column as a System.Int16 type.
public override unsafe int	GetInt32(int) [page 283]	Returns the value for the specified column as a System.Int32 type.
public override unsafe long	GetInt64(int) [page 284]	Returns the value for the specified column as a System.Int64 type.
public override string	GetName(int) [page 285]	Returns the name of the specified column.
public override unsafe int	GetOrdinal(string) [page 286]	Returns the column ID of the named column.
public unsafe int	GetRowCount(int) [page 287]	UL Ext: Returns the number of rows in the cursor, within threshold.
public override DataTable	GetSchemaTable() [page 288]	Returns a System.Data.DataTable value that describes the column metadata of the ULDataReader object.
public override unsafe String	GetString(int) [page 290]	Returns the value for the specified column as a System.String type.
public unsafe TimeSpan	GetTimeSpan(int) [page 291]	Returns the value for the specified column as a System.TimeSpan type with millisecond accuracy.
public unsafe ushort	GetUInt16(int) [page 292]	Returns the value for the specified column as a System.UInt16 type.
public unsafe uint	GetUInt32(int) [page 293]	Returns the value for the specified column as a System.UInt32 type.
public unsafe ulong	GetUInt64(int) [page 294]	Returns the value for the specified column as a System.UInt64 type.
public override object	GetValue(int) [page 295]	Returns the value of the specified column in its native format.
public override int	GetValues(object[]) [page 296]	Returns all the column values for the current row.

Modifier and Type	Method	Description
public override unsafe bool	IsDBNull(int) [page 297]	Checks whether the value from the specified column is NULL.
public void	MoveAfterLast() [page 298]	UL Ext: Positions the cursor to after the last row of the cursor.
public void	MoveBeforeFirst() [page 298]	UL Ext: Positions the cursor to before the first row of the cursor.
public unsafe bool	MoveFirst() [page 299]	UL Ext: Positions the cursor to the first row of the cursor.
public unsafe bool	MoveLast() [page 299]	UL Ext: Positions the cursor to the last row of the cursor.
public unsafe bool	MoveNext() [page 300]	UL Ext: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
public unsafe bool	MovePrevious() [page 301]	UL Ext: Positions the cursor to the previous row or before the first row.
public unsafe bool	MoveRelative(int) [page 301]	UL Ext: Positions the cursor relative to the current row.
public override bool	NextResult() [page 302]	Advances the ULDataReader object to the next result when reading the results of batch SQL statements.
public override bool	Read() [page 303]	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
protected void	Validate [page 304]	See individual topics

Properties

Modifier and Type	Property	Description
public override int	Depth [page 305]	Returns the depth of nesting for the current row.
public override int	FieldCount [page 305]	Returns the number of columns in the cursor.
public override unsafe bool	HasRows [page 306]	Checks whether the ULDataReader object has one or more rows.
public unsafe bool	IsBOF [page 306]	UL Ext: Checks whether the current row position is before the first row.
public override bool	IsClosed [page 307]	Checks whether the cursor is currently open.
public unsafe bool	IsEOF [page 307]	UL Ext: Checks whether the current row position is after the last row.
public override int	RecordsAffected [page 308]	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement.

Modifier and Type	Property	Description
public int	RowCount [page 308]	UL Ext: Returns the number of rows in the cursor.
public ULCursorSchema	Schema [page 309]	UL Ext: Holds the schema of this cursor.
public override object	this [page 310]	Returns the value of the specified column in its native format.

Remarks

Cursors are sets of rows from either a table or the result set from a query.

There is no constructor for the ULDataReader class. To get a ULDataReader object, execute a ULCommand object:

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
   "SELECT emp_id FROM employee", conn _
)
Dim reader As ULDataReader = cmd.ExecuteReader()
```

The following code is the C# language equivalent:

```
// C#
ULCommand cmd = new ULCommand(
   "SELECT emp_id FROM employee", conn
);
ULDataReader reader = cmd.ExecuteReader();
```

UL Ext: The ADO.NET standard only requires forward-only motion through the result set, but ULDataReader objects are bi-directional. ULDataReader's Move methods provide you with full flexibility when moving through results.

A ULDataReader object is a read-only result set. If you need a more flexible object to manipulate results, use the ULCommand.ExecuteResultSet method, the ULCommand.ExecuteTable method, or the ULDataAdapter class. The ULDataReader class retrieves rows as needed, whereas the ULDataAdapter class 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 class a much faster response time.

UL Ext: All columns of a ULDataReader object may be retrieved using the GetString method.

In this section:

Close() Method [page 266]
: Closes the cursor.

Dispose(bool) Method [page 267]

GetBoolean(int) Method [page 267]
: Returns the value for the specified column as a System.Boolean.

GetByte(int) Method [page 268]
: Returns the value for the specified column as an unsigned 8-bit value (System.Byte).

GetBytes Method [page 269]
UL Ext: Returns the value for the specified column as an array of System.Bytes values.

GetChar(int) Method [page 272]
This method is not supported in UltraLite.NET.

GetChars(int, long, char[], int, int) Method [page 273]
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.

GetDataTypeName(int) Method [page 274]
Returns the name of the specified column's provider data type.

GetDateTime(int) Method [page 275]
Returns the value for the specified column as a System.DateTime type with millisecond accuracy.

GetDbDataReader(int) Method [page 276]

GetDecimal(int) Method [page 276]
Returns the value for the specified column as a System.Decimal type.

GetDouble(int) Method [page 277]
Returns the value for the specified column as a System.Double type.

GetEnumerator() Method [page 278]
Returns an System.Collections.IEnumerator value that iterates through the ULDataReader object.

GetFieldType(int) Method [page 279]
Returns the System.Type value most appropriate for the specified column.

GetFloat(int) Method [page 280]
Returns the value for the specified column as a System.Single type.

GetGuid(int) Method [page 281]
Returns the value for the specified column as a UUID (System.Guid) type.

GetInt16(int) Method [page 282]
Returns the value for the specified column as a System.Int16 type.

GetInt32(int) Method [page 283]
Returns the value for the specified column as a System.Int32 type.

GetInt64(int) Method [page 284]
Returns the value for the specified column as a System.Int64 type.

GetName(int) Method [page 285]
Returns the name of the specified column.

GetOrdinal(string) Method [page 286]
Returns the column ID of the named column.

GetRowCount(int) Method [page 287]
UL Ext: Returns the number of rows in the cursor, within threshold.

GetSchemaTable() Method [page 288]
Returns a System.Data.DataTable value that describes the column metadata of the ULDataReader object.

GetString(int) Method [page 290]
Returns the value for the specified column as a System.String type.

GetTimeSpan(int) Method [page 291]

Returns the value for the specified column as a System.TimeSpan type with millisecond accuracy.

GetUInt16(int) Method [page 292]
Returns the value for the specified column as a System.UInt16 type.

GetUInt32(int) Method [page 293]
Returns the value for the specified column as a System.UInt32 type.

GetUInt64(int) Method [page 294]
Returns the value for the specified column as a System.UInt64 type.

GetValue(int) Method [page 295]
Returns the value of the specified column in its native format.

GetValues(object[]) Method [page 296]
Returns all the column values for the current row.

IsDBNull(int) Method [page 297]
Checks whether the value from the specified column is NULL.

MoveAfterLast() Method [page 298]
UL Ext: Positions the cursor to after the last row of the cursor.

MoveBeforeFirst() Method [page 298]
UL Ext: Positions the cursor to before the first row of the cursor.

MoveFirst() Method [page 299]
UL Ext: Positions the cursor to the first row of the cursor.

MoveLast() Method [page 299]
UL Ext: Positions the cursor to the last row of the cursor.

MoveNext() Method [page 300]
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 [page 301]
UL Ext: Positions the cursor to the previous row or before the first row.

MoveRelative(int) Method [page 301]
UL Ext: Positions the cursor relative to the current row.

NextResult() Method [page 302]
Advances the ULDataReader object to the next result when reading the results of batch SQL statements.

Read() Method [page 303]
Positions the cursor to the next row, or after the last row if the cursor was already on the last row.

Validate Method [page 304]

Depth Property [page 305]
Returns the depth of nesting for the current row.

FieldCount Property [page 305]
Returns the number of columns in the cursor.

HasRows Property [page 306]
Checks whether the ULDataReader object has one or more rows.

IsBOF Property [page 306]
UL Ext: Checks whether the current row position is before the first row.

IsClosed Property [page 307]
: Checks whether the cursor is currently open.

IsEOF Property [page 307]
: UL Ext: Checks whether the current row position is after the last row.

RecordsAffected Property [page 308]
: Returns the number of rows changed, inserted, or deleted by execution of the SQL statement.

RowCount Property [page 308]
: UL Ext: Returns the number of rows in the cursor.

Schema Property [page 309]
: UL Ext: Holds the schema of this cursor.

this Property [page 310]
: Returns the value of the specified column in its native format.

Related Information

ULCommand Class [page 45]
ExecuteResultSet() Method [page 76]
ExecuteTable() Method [page 81]
ULDataAdapter Class [page 224]
GetString(int) Method [page 290]

1.1.15.1 Close() Method

Closes the cursor.

Syntax

Visual Basic

```
Public Overrides Sub Close ()
```

C#

```
public override void Close ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

It is not an error to close a cursor that is already closed.

1.1.15.2 Dispose(bool) Method

Syntax

Visual Basic

```
Protected Overrides Sub Dispose (ByVal disposing As Boolean)
```

C#

```
protected override void Dispose (bool disposing)
```

1.1.15.3 GetBoolean(int) Method

Returns the value for the specified column as a System.Boolean.

Syntax

Visual Basic

```
Public Overrides Function GetBoolean (ByVal colID As Integer) As Boolean
```

C#

```
public override unsafe bool GetBoolean (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Boolean.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.4 GetByte(int) Method

Returns the value for the specified column as an unsigned 8-bit value (System.Byte).

Syntax

Visual Basic

```
Public Overrides Function GetByte (ByVal colID As Integer) As Byte
```

C#

```
public override unsafe byte GetByte (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Byte.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.5 GetBytes Method

UL Ext: Returns the value for the specified column as an array of System.Bytes values.

Overload List

Modifier and Type	Overload name	Description
public unsafe byte[]	GetBytes(int) [page 269]	UL Ext: Returns the value for the specified column as an array of System.Bytes values.
public override unsafe long	GetBytes(int, long, byte[], int, int) [page 270]	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.

In this section:

GetBytes(int) Method [page 269]
UL Ext: Returns the value for the specified column as an array of System.Bytes values.

GetBytes(int, long, byte[], int, int) Method [page 270]
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.

1.1.15.5.1 GetBytes(int) Method

UL Ext: Returns the value for the specified column as an array of System.Bytes values.

Syntax

Visual Basic

```
Public Function GetBytes (ByVal colID As Integer) As Byte()
```

C#

```
public unsafe byte[] GetBytes (int colID)
```

Parameters

colID 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.

Returns

The column value as an array of System.Bytes type.

Exceptions

ULException class A SQL error occurred.

Remarks

Only valid for columns of the ULDbType.Binary, ULDbType.LongBinary, or ULDbType.UniqueIdentifier types.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
GetBytes(int, long, byte[], int, int) Method [page 270]

1.1.15.5.2 GetBytes(int, long, byte[], int, int) 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 colID As Integer,
    ByVal srcOffset As Long,
    ByVal dst As Byte(),
    ByVal dstOffset As Integer,
    ByVal count As Integer
) As Long
```

C#

```
public override unsafe long GetBytes (
    int colID,
    long srcOffset,
    byte[] dst,
    int dstOffset,
    int count
)
```

Parameters

colID 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.

Returns

The actual number of bytes copied.

Exceptions

ULException class A SQL error occurred.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), the GetBytes method returns the length of the field in bytes.

The bytes at the *srcOffset* through *srcOffset* +*count* -1 positions of the value are copied into the *dstOffset* through *dstOffset* +*count* -1 positions, 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 are true, a ULException object 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 the *dst* length.

For other errors, a ULException object with the appropriate error code is thrown.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
GetBytes(int) Method [page 269]
ULException Class [page 312]
FieldCount Property [page 305]

1.1.15.6 GetChar(int) Method

This method is not supported in UltraLite.NET.

Syntax

Visual Basic

```
Public Overrides Function GetChar (ByVal colID As Integer) As Char
```

C#

```
public override char GetChar (int colID)
```

Parameters

colID 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.

Returns

This method is not supported in UltraLite.NET.

Exceptions

ULException class This method is not supported in UltraLite.NET.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
GetString(int) Method [page 290]
FieldCount Property [page 305]

1.1.15.7 GetChars(int, long, char[], int, int) 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 colID As Integer,
    ByVal srcOffset As Long,
    ByVal dst As Char(),
    ByVal dstOffset As Integer,
    ByVal count As Integer
) As Long
```

C#

```
public override unsafe long GetChars (
    int colID,
    long srcOffset,
    char[] dst,
    int dstOffset,
    int count
)
```

Parameters

colID 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.

Returns

The actual number of characters copied.

Exceptions

ULException class A SQL error occurred.

Remarks

If you pass a *dst* buffer that is a null reference (Nothing in Visual Basic), the GetChars method returns the length of the field in characters.

The characters at the *srcOffset* through *srcOffset* +*count* -1 positions of the value are copied into the *dstOffset* through *dstOffset* +*count* -1 positions, 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 instances are true, a ULException object with code ULSQLCode.SQLE_INVALID_PARAMETER constant is thrown and the destination is not modified:

- The *srcOffset* value is negative.
- The *dstOffset* value is negative.
- The *count* value is negative.
- The *dstOffset* +*count* value is greater than the *dst* length.

For other errors, a ULException object with the appropriate error code is thrown.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
ULException Class [page 312]
FieldCount Property [page 305]

1.1.15.8 GetDataTypeName(int) Method

Returns the name of the specified column's provider data type.

Syntax

Visual Basic

```
Public Overrides Function GetDataTypeName (ByVal colID As Integer) As
String
```

C#

```
public override string GetDataTypeName (int colID)
```

Parameters

colID 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.

Returns

A string corresponding to the column's ULDbType type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetColumnULDbType(int) Method [page 221]
FieldCount Property [page 305]
ULDbType Enumeration [page 568]

1.1.15.9 GetDateTime(int) Method

Returns the value for the specified column as a System.DateTime type with millisecond accuracy.

Syntax

Visual Basic

```
Public Overrides Function GetDateTime (ByVal colID As Integer) As Date
```

C#

```
public override unsafe DateTime GetDateTime (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.DateTime type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.10 GetDbDataReader(int) Method

Syntax

Visual Basic

```
Protected Overrides Function GetDbDataReader (ByVal i As Integer) As
DbDataReader
```

C#

```
protected override DbDataReader GetDbDataReader (int i)
```

1.1.15.11 GetDecimal(int) Method

Returns the value for the specified column as a System.Decimal type.

Syntax

Visual Basic

```
Public Overrides Function GetDecimal (ByVal colID As Integer) As Decimal
```

C#

```
public override decimal GetDecimal (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Decimal type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.12 GetDouble(int) Method

Returns the value for the specified column as a System.Double type.

Syntax

Visual Basic

```
Public Overrides Function GetDouble (ByVal colID As Integer) As Double
```

C#

```
public override unsafe double GetDouble (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Double type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.13 GetEnumerator() Method

Returns an System.Collections.IEnumerator value that iterates through the ULDataReader object.

Syntax

Visual Basic

```
Public Overrides Function GetEnumerator () As
System.Collections.IEnumerator
```

C#

```
public override IEnumerator GetEnumerator ()
```

Returns

A System.Collections.IEnumerator for the ULDataReader object.

1.1.15.14 GetFieldType(int) Method

Returns the System.Type value most appropriate for the specified column.

Syntax

Visual Basic

```
Public Overrides Function GetFieldType (ByVal colID As Integer) As Type
```

C#

```
public override Type GetFieldType (int colID)
```

Parameters

colID 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.

Returns

A System.Type value for the column.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetDataTypeName(int) Method [page 274]
GetColumnULDbType(int) Method [page 221]
FieldCount Property [page 305]

1.1.15.15 GetFloat(int) Method

Returns the value for the specified column as a System.Single type.

Syntax

Visual Basic

```
Public Overrides Function GetFloat (ByVal colID As Integer) As Single
```

C#

```
public override unsafe float GetFloat (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Single type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.16 GetGuid(int) Method

Returns the value for the specified column as a UUID (System.Guid) type.

Syntax

Visual Basic

```
Public Overrides Function GetGuid (ByVal colID As Integer) As Guid
```

C#

```
public override unsafe Guid GetGuid (int colID)
```

Parameters

colID 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.

Returns

The column value as a GUID type.

Exceptions

ULException class A SQL error occurred.

Remarks

This method is only valid for columns of the ULDbType.UniqueIdentifier type or the ULDbType.Binary type with length 16.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
GetColumnULDbType(int) Method [page 221]

GetColumnSize(int) Method [page 219]
FieldCount Property [page 305]

1.1.15.17 GetInt16(int) Method

Returns the value for the specified column as a System.Int16 type.

Syntax

Visual Basic

```
Public Overrides Function GetInt16 (ByVal colID As Integer) As Short
```

C#

```
public override unsafe short GetInt16 (int colID)
```

Parameters

colID 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.

Returns

The column value as an System.Int16 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.18 GetInt32(int) Method

Returns the value for the specified column as a System.Int32 type.

Syntax

Visual Basic

```
Public Overrides Function GetInt32 (ByVal colID As Integer) As Integer
```

C#

```
public override unsafe int GetInt32 (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.Int32 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.19 GetInt64(int) Method

Returns the value for the specified column as a System.Int64 type.

Syntax

Visual Basic

```
Public Overrides Function GetInt64 (ByVal colID As Integer) As Long
```

C#

```
public override unsafe long GetInt64 (int colID)
```

Parameters

colID 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.

Returns

The column value as an System.Int64 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.20 GetName(int) Method

Returns the name of the specified column.

Syntax

Visual Basic

```
Public Overrides Function GetName (ByVal colID As Integer) As String
```

C#

```
public override string GetName (int colID)
```

Parameters

colID 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.

Returns

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.

Exceptions

ULException class A SQL error occurred.

Remarks

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, the MyTable.ID value is the name of the only column in the result set for the "SELECT ID FROM MyTable" query.

This method is identical to the ULCursorSchema.GetColumnName method.

Related Information

FieldCount Property [page 305]

GetSchemaTable() Method [page 288]
GetColumnName(int) Method [page 216]
FieldCount Property [page 305]

1.1.15.21 GetOrdinal(string) 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 unsafe int GetOrdinal (string columnName)
```

Parameters

columnName The name of the column.

Returns

The column ID of the named column.

Exceptions

ULException class A SQL error occurred.

Remarks

Column IDs range from 0 to ULDataReader.FieldCount-1, inclusively.

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, the MyTable.ID value is the name of the only column in the result set for the "SELECT ID FROM MyTable" query.

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.

Related Information

GetSchemaTable() Method [page 288]
FieldCount Property [page 305]
GetColumnID(string) Method [page 215]

1.1.15.22 GetRowCount(int) Method

UL Ext: Returns the number of rows in the cursor, within threshold.

Syntax

Visual Basic

```
Public Function GetRowCount (ByVal threshold As Integer) As Integer
```

C#

```
public unsafe int GetRowCount (int threshold)
```

Parameters

threshold Threshold limit for row count.

Exceptions

ULException class A SQL error occurred.

Remarks

The number of rows in the cursor.

The RowCount property is expensive with complex queries, as it requires passing through the cursor rows. By using the GetRowCount(threshold) method, the caller can determine if there are at least threshold rows. If the

number of rows is below the threshold, that number is returned; otherwise, threshold is returned. This can be called again with a higher threshold.

If threshold is 0, returns the RowCount property.

Related Information

RowCount Property [page 308]

1.1.15.23 GetSchemaTable() Method

Returns a System.Data.DataTable value that describes the column metadata of the ULDataReader object.

Syntax

Visual Basic

```
Public Overrides Function GetSchemaTable () As DataTable
```

C#

```
public override DataTable GetSchemaTable ()
```

Returns

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. 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].

DataTable column	Description
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 the IsKey value 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 BaseTableName table 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 value 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.

DataTable column	Description
IsExpression	True if the column is an expression, false if it is a column value.

Related Information

Schema Property [page 309]
FieldCount Property [page 305]
ULDbType Enumeration [page 568]

1.1.15.24 GetString(int) Method

Returns the value for the specified column as a System.String type.

Syntax

Visual Basic

```
Public Overrides Function GetString (ByVal colID As Integer) As String
```

C#

```
public override unsafe String GetString (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.String type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.25 GetTimeSpan(int) Method

Returns the value for the specified column as a System.TimeSpan type with millisecond accuracy.

Syntax

Visual Basic

```
Public Function GetTimeSpan (ByVal colID As Integer) As TimeSpan
```

C#

```
public unsafe TimeSpan GetTimeSpan (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.TimeSpan type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]

FieldCount Property [page 305]

1.1.15.26 GetUInt16(int) Method

Returns the value for the specified column as a System.UInt16 type.

Syntax

Visual Basic

```
Public Function GetUInt16 (ByVal colID As Integer) As UShort
```

C#

```
public unsafe ushort GetUInt16 (int colID)
```

Parameters

colID 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.

Returns

The column value as an System.UInt16 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.27 GetUInt32(int) Method

Returns the value for the specified column as a System.UInt32 type.

Syntax

Visual Basic

```
Public Function GetUInt32 (ByVal colID As Integer) As UInteger
```

C#

```
public unsafe uint GetUInt32 (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.UInt32 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.28 GetUInt64(int) Method

Returns the value for the specified column as a System.UInt64 type.

Syntax

Visual Basic

```
Public Function GetUInt64 (ByVal colID As Integer) As ULong
```

C#

```
public unsafe ulong GetUInt64 (int colID)
```

Parameters

colID 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.

Returns

The column value as a System.UInt64 type.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.29 GetValue(int) Method

Returns the value of the specified column in its native format.

Syntax

Visual Basic

```
Public Overrides Function GetValue (ByVal colID As Integer) As Object
```

C#

```
public override object GetValue (int colID)
```

Parameters

colID 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.

Returns

The column value as the .NET type most appropriate for the column or the DBNull type if column is NULL.

Exceptions

ULException class A SQL error occurred.

Remarks

This method is identical to using the ULDataReader.this[int] method.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
FieldCount Property [page 305]

1.1.15.30 GetValues(object[]) 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.

Returns

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.

Exceptions

ArgumentNullException The *values* array is NULL or has zero length.
ULException class A SQL error occurred.

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 the DBNull type for NULL database columns. For other columns, it returns the value of the column in its native format.

Related Information

FieldCount Property [page 305]
GetFieldType(int) Method [page 279]
GetValue(int) Method [page 295]

1.1.15.31 IsDBNull(int) Method

Checks whether the value from the specified column is NULL.

Syntax

Visual Basic

```
Public Overrides Function IsDBNull (ByVal colID As Integer) As Boolean
```

C#

```
public override unsafe bool IsDBNull (int colID)
```

Parameters

colID 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.

Returns

True if value is NULL, false if value is not NULL.

Exceptions

ULException class A SQL error occurred.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]

FieldCount Property [page 305]

1.1.15.32 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.

1.1.15.33 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.

1.1.15.34 MoveFirst() Method

UL Ext: Positions the cursor to the first row of the cursor.

Syntax

Visual Basic

```
Public Function MoveFirst () As Boolean
```

C#

```
public unsafe bool MoveFirst ()
```

Returns

True if successful, false otherwise. For example, the method fails if there are no rows.

Exceptions

ULException class A SQL error occurred.

1.1.15.35 MoveLast() Method

UL Ext: Positions the cursor to the last row of the cursor.

Syntax

Visual Basic

```
Public Function MoveLast () As Boolean
```

C#

```
public unsafe bool MoveLast ()
```

Returns

True if successful, false otherwise. For example, the method fails if there are no rows.

Exceptions

ULException class A SQL error occurred.

1.1.15.36 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 unsafe bool MoveNext ()
```

Returns

True if successful, false otherwise. For example, the method fails if there are no more rows.

Exceptions

ULException class A SQL error occurred.

Remarks

This method is identical to the ULDataReader.Read method.

Related Information

Read() Method [page 303]

1.1.15.37 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 unsafe bool MovePrevious ()
```

Returns

True if successful, false otherwise. For example, the method fails if there are no more rows.

Exceptions

ULException class A SQL error occurred.

1.1.15.38 MoveRelative(int) 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 unsafe bool MoveRelative (int offset)
```

Parameters

offset The number of rows to move. Negative values correspond to moving backward.

Returns

True if successful, false otherwise. For example, the method fails if it positions beyond the first or last row.

Exceptions

ULException class A SQL error occurred.

Remarks

If the row does not exist, the method returns false, and the cursor position is after the last row (the ULDataReader.IsEOF method) if *offset* value is positive, and before the first row (the ULDataReader.IsBOF method) if the *offset* value is negative.

Related Information

IsEOF Property [page 307]
IsBOF Property [page 306]

1.1.15.39 NextResult() Method

Advances the ULDataReader object 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 ()
```

Returns

True if there are more result sets, false otherwise. For UltraLite.NET, always returns false.

Exceptions

ULException class The ULDataReader object is not opened.

Remarks

UL Ext: UltraLite.NET does not support batches of SQL statements, so the ULDataReader object is always positioned on the first and only result set. NextResult method calls have no effect.

1.1.15.40 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 ()
```

Returns

True if successful, false otherwise. For example, the method fails if there are no more rows.

Exceptions

ULException class A SQL error occurred.

Remarks

This method is identical to the ULDataReader.MoveNext method.

Related Information

MoveNext() Method [page 300]

1.1.15.41 Validate Method

Overload List

Modifier and Type	Overload name	Description
protected void	Validate() [page 304]	
protected void	Validate(int) [page 304]	

In this section:

Validate() Method [page 304]

Validate(int) Method [page 304]

1.1.15.41.1 Validate() Method

Syntax

Visual Basic

```
Protected Sub Validate ()
```

C#

```
protected void Validate ()
```

1.1.15.41.2 Validate(int) Method

Syntax

Visual Basic

```
Protected Sub Validate (ByVal colID As Integer)
```

C#

```
protected void Validate (int colID)
```

1.1.15.42 Depth Property

Returns the depth of nesting for the current row.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Depth As Integer
```

C#

```
public override int Depth {get;}
```

Remarks

The outermost table has a depth of zero.

All UltraLite.NET result sets have a depth of zero.

1.1.15.43 FieldCount Property

Returns the number of columns in the cursor.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property FieldCount As Integer
```

C#

```
public override int FieldCount {get;}
```

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.15.44 HasRows Property

Checks whether the ULDataReader object has one or more rows.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property HasRows As Boolean
```

C#

```
public override unsafe bool HasRows {get;}
```

Remarks

True if the result set has at least one row, false if there are no rows.

1.1.15.45 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 unsafe bool IsBOF {get;}
```

Remarks

True if the current row position is before the first row, false otherwise.

1.1.15.46 IsClosed Property

Checks whether the cursor is currently open.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property IsClosed As Boolean
```

C#

```
public override bool IsClosed {get;}
```

Remarks

True if the cursor is currently open, false if the cursor is closed.

1.1.15.47 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 unsafe bool IsEOF {get;}
```

Remarks

True if the current row position is after the last row, false otherwise.

1.1.15.48 RecordsAffected Property

Returns the number of rows changed, inserted, or deleted by execution of the SQL statement.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property RecordsAffected As Integer
```

C#

```
public override int RecordsAffected {get;}
```

Remarks

For SELECT statements or CommandType.TableDirect tables, this value is -1.

The number of rows changed, inserted, or deleted by execution of the SQL statement.

1.1.15.49 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;}
```

Remarks

The number of rows in the cursor.

You can use the RowCount method 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.

Related Information

StartSynchronizationDelete() Method [page 155]
StopSynchronizationDelete() Method [page 156]

1.1.15.50 Schema Property

UL Ext: Holds the schema of this cursor.

Syntax

Visual Basic

```
Public ReadOnly Property Schema As ULCursorSchema
```

C#

```
public ULCursorSchema Schema {get;}
```

Remarks

For result sets, the ULResultSetSchema object representing the schema of the result set. For tables, the ULTableSchema object representing the schema of the table.

This property represents the complete schema of the cursor, including UltraLite.NET extended information which is not represented in the results from the ULDataReader.GetSchemaTable method.

Related Information

ULTableSchema Class [page 537]
GetSchemaTable() Method [page 288]
ULResultSetSchema Class [page 453]

1.1.15.51 this Property

Returns the value of the specified column in its native format.

Overload List

Modifier and Type	Overload name	Description
public override object	this[int colID] [page 310]	Returns the value of the specified column in its native format.
public override object	this[string name] [page 311]	Returns the value of the specified named column in its native format.

In this section:

this[int colID] Property [page 310]
Returns the value of the specified column in its native format.

this[string name] Property [page 311]
Returns the value of the specified named column in its native format.

1.1.15.51.1 this[int colID] Property

Returns the value of the specified column in its native format.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Item (ByVal colIDAs Integer) As Object
```

C#

```
public override object this[int colID] {get;}
```

Returns

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

In C#, this property is the indexer for the ULDataReader class.

This method is identical in functionality to the ULDataReader.GetValue(int) method.

Related Information

GetFieldType(int) Method [page 279]
GetValue(int) Method [page 295]
FieldCount Property [page 305]

1.1.15.51.2 this[string name] Property

Returns the value of the specified named column in its native format.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Item (ByVal name As String) As Object
```

C#

```
public override object this[string name] {get;}
```

Returns

The column value as the .NET type most appropriate for the column or DBNull if column is NULL.

Remarks

In C#, this property is the indexer for the ULDataReader object.

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, the MyTable.ID value 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 ) )
```

Related Information

GetOrdinal(string) Method [page 286]
GetValue(int) Method [page 295]
GetFieldType(int) Method [page 279]

1.1.16 ULException Class

Represents a SQL error returned by the UltraLite.NET database.

Syntax

Visual Basic

```
Public NotInheritable Class ULException Inherits
System.ApplicationException
```

C#

```
public sealed class ULException : System.ApplicationException
```

Members

All members of ULException, including inherited members.

Properties

Modifier and Type	Property	Description
public ULSQLCode	NativeError [page 313]	Returns the SQLCODE value returned by the database.
public override string	Source [page 313]	Returns the name of the provider that generated the error.

Remarks

The SQLCODE denoting the error is returned in the NativeError property.

This class is not serializable under the .NET Compact Framework.

In this section:

Related Information

NativeError Property [page 313]

1.1.16.1 NativeError Property

Returns the SQLCODE value returned by the database.

Syntax

Visual Basic

```
Public ReadOnly Property NativeError As ULSQLCode
```

C#

```
public ULSQLCode NativeError {get;}
```

Remarks

The ULSQLCode value returned by the database.

1.1.16.2 Source Property

Returns the name of the provider that generated the error.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Source As String
```

C#

```
public override string Source {get;}
```

Remarks

The string value identifying UltraLite.NET as the provider.

1.1.17 ULFactory Class

Represents a set of methods for creating instances of the Sap.Data.UltraLite provider's implementation of the data source classes.

Syntax

Visual Basic

```
Public Class ULFactory Inherits System.Data.Common.DbProviderFactory
```

C#

```
public class ULFactory : System.Data.Common.DbProviderFactory
```

Members

All members of ULFactory, including inherited members.

Variables

Modifier and Type	Variable	Description
public static readonly ULFactory	Instance	Represents the singleton instance of the ULFactory class. The ULFactory class is not available in the .NET Compact Framework 2.0. 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 the ULFactory class using the System.Data.Common.DbProviderFactories.GetFactory(String) method.

Methods

Modifier and Type	Method	Description
public override DbCommand	CreateCommand() [page 316]	Returns a strongly typed System.Data.Common.DbCommand instance.
public override DbCommandBuilder	CreateCommandBuilder() [page 317]	Returns a strongly typed System.Data.Common.DbCommandBuilder instance.
public override DbConnection	CreateConnection() [page 317]	Returns a strongly typed System.Data.Common.DbConnection instance.
public override DbConnectionStringBuilder	CreateConnectionStringBuilder() [page 318]	Returns a strongly typed System.Data.Common.DbConnectionStringBuilder instance.
public override DbDataAdapter	CreateDataAdapter() [page 318]	Returns a strongly typed System.Data.Common.DbDataAdapter instance.
public override DbParameter	CreateParameter() [page 319]	Returns a strongly typed System.Data.Common.DbParameter instance.

Properties

Modifier and Type	Property	Description
public override bool	CanCreateDataSourceEnumerator [page 319]	Returns false to indicate the UltraLite.NET does not support the DbDataSourceEnumerator class.

Remarks

The ULFactory class is not available in the .NET Compact Framework 2.0.

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 Sap.Data.UltraLite as the provider invariant name passed to the GetFactory method. For example:

```
' Visual Basic
Dim factory As DbProviderFactory =
   DbProviderFactories.GetFactory( "Sap.Data.UltraLite" )
Dim conn As DbConnection =
   factory.CreateConnection()
```

The following code is the C# language equivalent:

```
// C#
DbProviderFactory factory =
   DbProviderFactories.GetFactory( "Sap.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 . UltraLite.NET does not support the CreateCommandBuilder, CreateDataSourceEnumerator, and CreatePermission methods. Custom Attribute: sealed

In this section:

1.1.17.1 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 ()
```

Returns

A new ULCommand instance typed as DbCommand.

Related Information

ULCommand Class [page 45]

1.1.17.2 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 ()
```

Returns

A new ULCommandBuilder instance typed as DbCommandBuilder.

Related Information

ULCommandBuilder Class [page 92]

1.1.17.3 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 ()
```

Returns

A new ULConnection instance typed as DbConnection.

Related Information

ULConnection Class [page 108]

1.1.17.4 CreateConnectionStringBuilder() Method

Returns a strongly typed System.Data.Common.DbConnectionStringBuilder instance.

Syntax

Visual Basic

```
Public Overrides Function CreateConnectionStringBuilder () As
DbConnectionStringBuilder
```

C#

```
public override DbConnectionStringBuilder CreateConnectionStringBuilder ()
```

Returns

A new ULConnectionStringBuilder instance typed as DbConnectionStringBuilder.

Related Information

ULConnectionStringBuilder Class [page 183]

1.1.17.5 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 ()
```

Returns

A new ULDataAdapter instance typed as DbDataAdapter.

Related Information

ULDataAdapter Class [page 224]

1.1.17.6 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 ()
```

Returns

A new ULParameter instance typed as DbParameter.

Related Information

ULParameter Class [page 372]

1.1.17.7 CanCreateDataSourceEnumerator Property

Returns false to indicate the UltraLite.NET does not support the DbDataSourceEnumerator class.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property CanCreateDataSourceEnumerator As Boolean
```

C#

```
public override bool CanCreateDataSourceEnumerator {get;}
```

Remarks

False to indicate that the ULFactory class does not implement the CreateDataSourceEnumerator method.

1.1.18 ULFileTransfer Class

UL Ext: Transfers a file from a remote database using the MobiLink server.

Syntax

Visual Basic

```
Public NotInheritable Class ULFileTransfer
```

C#

```
public sealed class ULFileTransfer
```

Members

All members of ULFileTransfer, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULFileTransfer() [page 323]	Initializes a ULFileTransfer object.

Methods

Modifier and Type	Method	Description
public bool	DownloadFile [page 324]	Download the file specified by the properties of this object.
public bool	UploadFile [page 327]	Upload the file specified by the properties of this object.

Properties

Modifier and Type	Property	Description
public unsafe String[]	AuthenticationParms [page 330]	Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).
public ULAuthStatusCode	AuthStatus [page 331]	Returns the authorization status code for the last file transfer attempt.
public long	AuthValue [page 331]	Returns the return value from custom user authentication synchronization scripts.
public ushort	FileAuthCode [page 332]	Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.
public string	FileName [page 332]	Specifies the name of the file to download.
public string	LocalFileName [page 333]	Specifies the local file name for the downloaded file.
public string	LocalPath [page 334]	Specifies where to download the file.
public string	Password [page 334]	The MobiLink password for the user specified by the UserName value.
public string	RemoteKey [page 335]	The key that uniquely identifies the MobiLink client to the MobiLink server.
public bool	ResumePartialDownload [page 336]	Specifies whether to resume or discard a previous partial download.
public ULStreamType	Stream [page 336]	Specifies the MobiLink synchronization stream to use for the file transfer.
public ULStreamErrorCode	StreamErrorCode [page 337]	Returns the error reported by the stream itself for the last file transfer attempt.
public int	StreamErrorSystem [page 338]	Returns the stream error system-specific code.
public string	StreamParms [page 338]	Specifies the parameters to configure the synchronization stream.
public bool	TransferredFile [page 339]	Checks whether the file was actually downloaded during the last file transfer attempt.
public string	UserName [page 340]	The user name that identifies the MobiLink client to the MobiLink server.
public string	Version [page 340]	Specifies which synchronization script to use.

Remarks

You do not need a database connection to perform a file transfer, however, if your application uses an UltraLite database with the UltraLite Engine runtime, you must set the ULDatabaseManager.RuntimeType value 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 values.

In this section:

StreamParms Property [page 338]
Specifies the parameters to configure the synchronization stream.

TransferredFile Property [page 339]
Checks whether the file was actually downloaded during the last file transfer attempt.

UserName Property [page 340]
The user name that identifies the MobiLink client to the MobiLink server.

Version Property [page 340]
Specifies which synchronization script to use.

Related Information

FileName Property [page 332]
Stream Property [page 336]
UserName Property [page 340]
Version Property [page 340]

1.1.18.1 ULFileTransfer() Constructor

Initializes a ULFileTransfer object.

Syntax

Visual Basic

```
Public Sub ULFileTransfer ()
```

C#

```
public ULFileTransfer ()
```

Remarks

The connection must be opened before you can perform any operations against the database.

You do not need a database connection to perform a file transfer, however, if your application uses an UltraLite database with the UltraLite Engine runtime, you must set the ULDatabaseManager.RuntimeType value 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 values set before it can transfer a file.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]
FileName Property [page 332]
Stream Property [page 336]
UserName Property [page 340]
Version Property [page 340]

1.1.18.2 DownloadFile Method

Download the file specified by the properties of this object.

Overload List

Modifier and Type	Overload name	Description
public bool	DownloadFile() [page 324]	Download the file specified by the properties of this object.
public bool	DownloadFile(ULFileTransferProgressListener) [page 326]	Download the file specified by the properties of this object with progress events posted to the specified listener.

In this section:

DownloadFile() Method [page 324]
Download the file specified by the properties of this object.

DownloadFile(ULFileTransferProgressListener) Method [page 326]
Download the file specified by the properties of this object with progress events posted to the specified listener.

1.1.18.2.1 DownloadFile() Method

Download the file specified by the properties of this object.

Syntax

Visual Basic

```
Public Function DownloadFile () As Boolean
```

C#

```
public bool DownloadFile ()
```

Returns

True if successful, false otherwise (check the ULFileTransfer.StreamErrorCode value and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName value is downloaded by the MobiLink server to the ULFileTransfer.LocalPath value using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version values.

To avoid file corruption, UltraLite.NET downloads to a temporary file and only replaces the local file once the download has completed. Other properties that affect the download are: ULFileTransfer.LocalFileName, ULFileTransfer.AuthenticationParms, and ULFileTransfer.ResumePartialDownload.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.TransferredFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem values.

Related Information

DownloadFile(ULFileTransferProgressListener) Method [page 326]
FileName Property [page 332]
LocalPath Property [page 334]
Stream Property [page 336]
UserName Property [page 340]
Password Property [page 334]
Version Property [page 340]
LocalFileName Property [page 333]
AuthenticationParms Property [page 330]
ResumePartialDownload Property [page 336]
AuthStatus Property [page 331]
AuthValue Property [page 331]
FileAuthCode Property [page 332]
TransferredFile Property [page 339]
StreamErrorCode Property [page 337]
StreamErrorSystem Property [page 338]

StreamErrorCode Property [page 337]
UploadFile() Method [page 328]

1.1.18.2.2 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.

Returns

True if successful, false otherwise (check the ULFileTransfer.StreamErrorCode value and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName value is downloaded by the MobiLink server to the ULFileTransfer.LocalPath value using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version values.

To avoid file corruption, UltraLite.NET downloads to a temporary file and only replaces the local file once the download has completed. Other properties that affect the download are: ULFileTransfer.LocalFileName, ULFileTransfer.AuthenticationParms, and ULFileTransfer.ResumePartialDownload.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.TransferredFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem values.

Errors may result in no data being sent to the listener.

Related Information

FileName Property [page 332]
LocalPath Property [page 334]
Stream Property [page 336]
UserName Property [page 340]
Password Property [page 334]
Version Property [page 340]
LocalFileName Property [page 333]
AuthenticationParms Property [page 330]
ResumePartialDownload Property [page 336]
AuthStatus Property [page 331]
AuthValue Property [page 331]
FileAuthCode Property [page 332]
TransferredFile Property [page 339]
StreamErrorCode Property [page 337]
StreamErrorSystem Property [page 338]
StreamErrorCode Property [page 337]
UploadFile() Method [page 328]

1.1.18.3 UploadFile Method

Upload the file specified by the properties of this object.

Overload List

Modifier and Type	Overload name	Description
public bool	UploadFile() [page 328]	Upload the file specified by the properties of this object.
public bool	UploadFile(ULFileTransferProgressListener) [page 329]	Upload the file specified by the properties of this object with progress events posted to the specified listener.

In this section:

UploadFile() Method [page 328]
Upload the file specified by the properties of this object.

UploadFile(ULFileTransferProgressListener) Method [page 329]
Upload the file specified by the properties of this object with progress events posted to the specified listener.

1.1.18.3.1 UploadFile() Method

Upload the file specified by the properties of this object.

Syntax

Visual Basic

```
Public Function UploadFile () As Boolean
```

C#

```
public bool UploadFile ()
```

Returns

True if successful, false otherwise (check the ULFileTransfer.StreamErrorCode value and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName value is uploaded to the MobiLink server from the ULFileTransfer.LocalPath value using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version values.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.TransferredFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem values.

Related Information

UploadFile(ULFileTransferProgressListener) Method [page 329]
FileName Property [page 332]
LocalPath Property [page 334]
Stream Property [page 336]
UserName Property [page 340]
Password Property [page 334]
Version Property [page 340]
LocalFileName Property [page 333]
AuthenticationParms Property [page 330]
ResumePartialDownload Property [page 336]
AuthStatus Property [page 331]

AuthValue Property [page 331]
FileAuthCode Property [page 332]
TransferredFile Property [page 339]
StreamErrorCode Property [page 337]
StreamErrorSystem Property [page 338]
StreamErrorCode Property [page 337]

1.1.18.3.2 UploadFile(ULFileTransferProgressListener) Method

Upload the file specified by the properties of this object with progress events posted to the specified listener.

Syntax

Visual Basic

```
Public Function UploadFile (ByVal listener As
ULFileTransferProgressListener) As Boolean
```

C#

```
public bool UploadFile (ULFileTransferProgressListener listener)
```

Parameters

listener The object that receives file transfer progress events.

Returns

True if successful, false otherwise (check the ULFileTransfer.StreamErrorCode value and other status properties for reason).

Remarks

The file specified by the ULFileTransfer.FileName value is uploaded to the MobiLink server from the ULFileTransfer.LocalPath value using the ULFileTransfer.Stream, ULFileTransfer.UserName, ULFileTransfer.Password, and ULFileTransfer.Version values.

A detailed result status is reported in this object's ULFileTransfer.AuthStatus, ULFileTransfer.AuthValue, ULFileTransfer.FileAuthCode, ULFileTransfer.TransferredFile, ULFileTransfer.StreamErrorCode, and ULFileTransfer.StreamErrorSystem values.

Errors may result in no data being sent to the listener.

Related Information

UploadFile() Method [page 328]
FileName Property [page 332]
LocalPath Property [page 334]
Stream Property [page 336]
UserName Property [page 340]
Password Property [page 334]
Version Property [page 340]
LocalFileName Property [page 333]
AuthenticationParms Property [page 330]
ResumePartialDownload Property [page 336]
AuthStatus Property [page 331]
AuthValue Property [page 331]
FileAuthCode Property [page 332]
TransferredFile Property [page 339]
StreamErrorCode Property [page 337]
StreamErrorSystem Property [page 338]
StreamErrorCode Property [page 337]

1.1.18.4 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 unsafe String[] AuthenticationParms {get;set;}
```

Remarks

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.

Only the first 255 strings are used and each string should be no longer than the MobiLink server's limit for authentication parameters (currently 4000 UTF8 bytes).

1.1.18.5 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;}
```

Remarks

One of the ULAuthStatusCode values denoting the authorization status for the last file transfer attempt.

Related Information

ULAuthStatusCode Enumeration [page 566]

1.1.18.6 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;}
```

Remarks

A long integer returned from custom user authentication synchronization scripts.

1.1.18.7 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 UShort
```

C#

```
public ushort FileAuthCode {get;}
```

Remarks

An unsigned short integer returned from the authenticate_file_transfer script for the last file transfer attempt.

1.1.18.8 FileName Property

Specifies the name of the file to download.

Syntax

Visual Basic

```
Public Property FileName As String
```

C#

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

Remarks

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.

The FileName value is the name of the file on the server running MobiLink. MobiLink first searches 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). The FileName value must not include any drive or path information so that the MobiLink server can find it. For example, "myfile.txt" is valid, but "somedir\myfile.txt", "..\myfile.txt", and "c:\myfile.txt" are all invalid.

Related Information

LocalFileName Property [page 333]
DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.9 LocalFileName Property

Specifies the local file name for the downloaded file.

Syntax

Visual Basic

```
Public Property LocalFileName As String
```

C#

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

Remarks

A string specifying the local file name for the downloaded file. The FileName value is used if the value is a null reference (Nothing in Visual Basic). The default is a null reference (Nothing in Visual Basic).

Related Information

FileName Property [page 332]

1.1.18.10 LocalPath Property

Specifies where to download the file.

Syntax

Visual Basic

```
Public Property LocalPath As String
```

C#

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

Remarks

A string specifying the local directory of the file. The default is a null reference (Nothing in Visual Basic).

The default local directory varies depending on the device's operating system:

- For Windows Mobile devices, if the LocalPath value is a null reference (Nothing in Visual Basic), the file is stored in the root (\) directory.
- For desktop applications, if the LocalPath value is a null reference (Nothing in Visual Basic), the file is stored in the current directory.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.11 Password Property

The MobiLink password for the user specified by the UserName value.

Syntax

Visual Basic

```
Public Property Password As String
```

C#

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

Remarks

A string specifying the MobiLink password. The default is a null reference (Nothing in Visual Basic), meaning no password is specified.

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.

Related Information

UserName Property [page 340]
DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.12 RemoteKey Property

The key that uniquely identifies the MobiLink client to the MobiLink server.

Syntax

Visual Basic

```
Public Property RemoteKey As String
```

C#

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

Remarks

A string specifying the remote key. This property has no default value, and must be explicitly set.

The MobiLink server passes this value to various scripts to uniquely identify this client.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.13 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;}
```

Remarks

True to resume a previous partial download, false to discard a previous partial download. The default is false.

UltraLite.NET can use the ULFileTransferListener object to restart downloads that fail because of communication errors or user aborts. 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 is discarded and a new download starts.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.14 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;}
```

Remarks

One of the ULStreamType values specifying the type of synchronization stream to use. The default is the ULStreamType.TCPIP value.

Most synchronization streams require parameters to identify the MobiLink server address and control other behavior. These parameters are supplied by the ULFileTransfer.StreamParms value.

If the stream type is set to a value that is invalid for the platform, the stream type is set to the ULStreamType.TCPIP value.

Related Information

ULStreamType Enumeration [page 574]
StreamParms Property [page 338]
DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.15 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;}
```

Remarks

One of the ULStreamErrorCode values denoting the error reported by the stream itself. The ULStreamErrorCode.NONE value if no error occurred.

1.1.18.16 StreamErrorSystem Property

Returns the stream error system-specific code.

Syntax

Visual Basic

```
Public ReadOnly Property StreamErrorSystem As Integer
```

C#

```
public int StreamErrorSystem {get;}
```

Remarks

An integer denoting the stream error system-specific code.

1.1.18.17 StreamParms Property

Specifies the parameters to configure the synchronization stream.

Syntax

Visual Basic

```
Public Property StreamParms As String
```

C#

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

Remarks

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)

The StreamParms value 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").

Related Information

Stream Property [page 336]
DownloadFile() Method [page 324]
UploadFile() Method [page 328]
ULStreamType Enumeration [page 574]

1.1.18.18 TransferredFile Property

Checks whether the file was actually downloaded during the last file transfer attempt.

Syntax

Visual Basic

```
Public ReadOnly Property TransferredFile As Boolean
```

C#

```
public bool TransferredFile {get;}
```

Remarks

True if the file was downloaded, false otherwise.

If the file is already up-to-date when the DownloadFile or UploadFile method is invoked, it returns true, but TransferredFile is false. If an error occurs and the DownloadFile or UploadFile method returns false, then TransferredFile is false.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.19 UserName Property

The user name that identifies the MobiLink client to the MobiLink server.

Syntax

Visual Basic

```
Public Property UserName As String
```

C#

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

Remarks

A string specifying the user name. This property has no default value, and must be explicitly set.

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.

Related Information

Password Property [page 334]
DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.18.20 Version Property

Specifies which synchronization script to use.

Syntax

Visual Basic

```
Public Property Version As String
```

C#

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

Remarks

A string specifying the version of the synchronization script to use. This property has no default value, and must be explicitly set.

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.

Related Information

DownloadFile() Method [page 324]
UploadFile() Method [page 328]

1.1.19 ULFileTransferProgressData Class

UL Ext: Returns file transfer progress monitoring data.

Syntax

Visual Basic

```
Public Class ULFileTransferProgressData
```

C#

```
public class ULFileTransferProgressData
```

Members

All members of ULFileTransferProgressData, including inherited members.

Variables

Modifier and Type	Variable	Description
public const int	FLAG_IS_BLOCKING	A flag indicating that the file transfer is blocked awaiting a response from the MobiLink server.

Properties

Modifier and Type	Property	Description
public ulong	BytesReceived [page 342]	Returns the number of bytes received so far.

Modifier and Type	Property	Description
public ulong	FileSize [page 343]	Returns the size of the file being transferred.
public int	Flags [page 343]	Returns the current file transfer flags indicating additional information relating to the current state.
public ulong	ResumedAtSize [page 344]	Returns the point in the file where the transfer was resumed.

In this section:

BytesReceived Property [page 342]
Returns the number of bytes received so far.

FileSize Property [page 343]
Returns the size of the file being transferred.

Flags Property [page 343]
Returns the current file transfer flags indicating additional information relating to the current state.

ResumedAtSize Property [page 344]
Returns the point in the file where the transfer was resumed.

Related Information

ULFileTransferProgressListener Interface [page 344]

1.1.19.1 BytesReceived Property

Returns the number of bytes received so far.

Syntax

Visual Basic

```
Public ReadOnly Property BytesReceived As ULong
```

C#

```
public ulong BytesReceived {get;}
```

Remarks

The number of bytes received so far.

1.1.19.2 FileSize Property

Returns the size of the file being transferred.

Syntax

Visual Basic

```
Public ReadOnly Property FileSize As ULong
```

C#

```
public ulong FileSize {get;}
```

Remarks

The size of the file in bytes.

1.1.19.3 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;}
```

Remarks

An integer containing a combination of flags or'ed together.

1.1.19.4 ResumedAtSize Property

Returns the point in the file where the transfer was resumed.

Syntax

Visual Basic

```
Public ReadOnly Property ResumedAtSize As ULong
```

C#

```
public ulong ResumedAtSize {get;}
```

Remarks

The number of bytes transferred previously.

1.1.20 ULFileTransferProgressListener Interface

UL Ext: The listener interface for receiving file transfer progress events.

Syntax

Visual Basic

```
Public Interface ULFileTransferProgressListener
```

C#

```
public interface ULFileTransferProgressListener
```

Members

All members of ULFileTransferProgressListener, including inherited members.

Methods

Modifier and Type	Method	Description
public bool	FileTransferProgressed(ULFileTransfer-ProgressData) [page 345]	Invoked during a file transfer to inform the user of progress.

In this section:

FileTransferProgressed(ULFileTransferProgressData) Method [page 345]
Invoked during a file transfer to inform the user of progress.

Related Information

DownloadFile(ULFileTransferProgressListener) Method [page 326]

1.1.20.1 FileTransferProgressed(ULFileTransferProgressData) Method

Invoked during a file transfer to inform the user of progress.

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.

Returns

This method should return true to cancel the transfer or return false to continue.

Remarks

This method should return true to cancel the transfer or return false to continue.

No UltraLite.NET API methods should be invoked during a FileTransferProgressed call.

Related Information

ULFileTransferProgressData Class [page 341]

1.1.21 ULIndexSchema Class

UL Ext: Represents the schema of an UltraLite table index.

Syntax

Visual Basic

```
Public NotInheritable Class ULIndexSchema
```

C#

```
public sealed class ULIndexSchema
```

Members

All members of ULIndexSchema, including inherited members.

Methods

Modifier and Type	Method	Description
public void	Close() [page 348]	Closes the ULIndexSchema object.
public unsafe string	GetColumnName(short) [page 348]	Returns the name of the colOrdinalInIndex 'th column in this index.
public unsafe bool	IsColumnDescending(string) [page 349]	Checks whether the named column is used in descending order by the index.

Properties

Modifier and Type	Property	Description
public unsafe short	ColumnCount [page 350]	Returns the number of columns in the index.
public unsafe bool	IsForeignKey [page 351]	Checks whether the index is a foreign key.
public unsafe bool	IsForeignKeyCheckOnCommit [page 351]	Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.
public unsafe bool	IsForeignKeyNullable [page 352]	Checks whether the foreign key is nullable.

Modifier and Type	Property	Description
public bool	IsOpen [page 352]	Determines whether the index schema is open or closed.
public unsafe bool	IsPrimaryKey [page 353]	Checks whether the index is the primary key.
public unsafe bool	IsUniqueIndex [page 353]	Checks whether the index is unique.
public unsafe bool	IsUniqueKey [page 354]	Checks whether the index is a unique key.
public unsafe string	Name [page 354]	Returns the name of the index.
public unsafe string	ReferencedIndexName [page 355]	The name of the referenced primary index if the index is a foreign key.
public unsafe string	ReferencedTableName [page 355]	The name of the referenced primary table if the index is a foreign key.

Remarks

There is no constructor for this class. Index schemas are created using the ULTableSchema.PrimaryKey, ULTableSchema.GetIndex(string), and ULTableSchema.GetOptimalIndex(int) methods.

In this section:

Close() Method [page 348]
Closes the ULIndexSchema object.

GetColumnName(short) Method [page 348]
Returns the name of the colOrdinalInIndex 'th column in this index.

IsColumnDescending(string) Method [page 349]
Checks whether the named column is used in descending order by the index.

ColumnCount Property [page 350]
Returns the number of columns in the index.

IsForeignKey Property [page 351]
Checks whether the index is a foreign key.

IsForeignKeyCheckOnCommit Property [page 351]
Checks whether referential integrity for the foreign key is performed on commits or on inserts and updates.

IsForeignKeyNullable Property [page 352]
Checks whether the foreign key is nullable.

IsOpen Property [page 352]
Determines whether the index schema is open or closed.

IsPrimaryKey Property [page 353]
Checks whether the index is the primary key.

IsUniqueIndex Property [page 353]
Checks whether the index is unique.

IsUniqueKey Property [page 354]
: Checks whether the index is a unique key.

Name Property [page 354]
: Returns the name of the index.

ReferencedIndexName Property [page 355]
: The name of the referenced primary index if the index is a foreign key.

ReferencedTableName Property [page 355]
: The name of the referenced primary table if the index is a foreign key.

Related Information

PrimaryKey Property [page 556]
GetIndex(string) Method [page 543]
GetOptimalIndex(int) Method [page 544]
ULTableSchema Class [page 537]

1.1.21.1 Close() Method

Closes the ULIndexSchema object.

Syntax

Visual Basic

```
Public Sub Close ()
```

C#

```
public void Close ()
```

1.1.21.2 GetColumnName(short) 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 unsafe string GetColumnName (short colOrdinalInIndex)
```

Parameters

colOrdinalInIndex The ordinal of the desired column in the index. The value must be in the range [1,ColumnCount].

Returns

The name of the column.

Exceptions

ULException class A SQL error occurred.

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.

Related Information

ColumnCount Property [page 350]
ColumnCount Property [page 350]

1.1.21.3 IsColumnDescending(string) 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 unsafe bool IsColumnDescending (string name)
```

Parameters

name The name of the column.

Returns

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.

Related Information

GetColumnName(short) Method [page 348]
ColumnCount Property [page 350]

1.1.21.4 ColumnCount Property

Returns the number of columns in the index.

Syntax

Visual Basic

```
Public ReadOnly Property ColumnCount As Short
```

C#

```
public unsafe short ColumnCount {get;}
```

Remarks

The number of columns in the index.

Column ordinals in indexes range from 1 to the ColumnCount value, inclusively.

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.

1.1.21.5 IsForeignKey Property

Checks whether the index is a foreign key.

Syntax

Visual Basic

```
Public ReadOnly Property IsForeignKey As Boolean
```

C#

```
public unsafe bool IsForeignKey {get;}
```

Remarks

True if the index is the foreign key, false if the index is not the foreign key.

Columns in a foreign key may reference another table's non-null, unique index.

1.1.21.6 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 unsafe bool IsForeignKeyCheckOnCommit {get;}
```

Remarks

True if referential integrity is checked on commits, false if it is checked on inserts and updates.

Related Information

IsForeignKey Property [page 351]

1.1.21.7 IsForeignKeyNullable Property

Checks whether the foreign key is nullable.

Syntax

Visual Basic

```
Public ReadOnly Property IsForeignKeyNullable As Boolean
```

C#

```
public unsafe bool IsForeignKeyNullable {get;}
```

Remarks

True if the foreign key is nullable, false if the foreign key is not nullable.

Related Information

IsForeignKey Property [page 351]

1.1.21.8 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;}
```

Remarks

True if the index schema is open, otherwise false.

1.1.21.9 IsPrimaryKey Property

Checks whether the index is the primary key.

Syntax

Visual Basic

```
Public ReadOnly Property IsPrimaryKey As Boolean
```

C#

```
public unsafe bool IsPrimaryKey {get;}
```

Remarks

True if the index is the primary key, false if the index is not the primary key.

Columns in the primary key may not be null.

1.1.21.10 IsUniqueIndex Property

Checks whether the index is unique.

Syntax

Visual Basic

```
Public ReadOnly Property IsUniqueIndex As Boolean
```

C#

```
public unsafe bool IsUniqueIndex {get;}
```

Remarks

True if the index is unique, false if the index is not unique.

Columns in a unique index may be null.

1.1.21.11 IsUniqueKey Property

Checks whether the index is a unique key.

Syntax

Visual Basic

```
Public ReadOnly Property IsUniqueKey As Boolean
```

C#

```
public unsafe bool IsUniqueKey {get;}
```

Remarks

True if the index is a unique key, false if the index is not a unique key.

Columns in a unique key may not be null.

1.1.21.12 Name Property

Returns the name of the index.

Syntax

Visual Basic

```
Public ReadOnly Property Name As String
```

C#

```
public unsafe string Name {get;}
```

Remarks

A string specifying the name of the index.

1.1.21.13 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 unsafe string ReferencedIndexName {get;}
```

Remarks

A string specifying the name of the referenced primary index.

Related Information

IsForeignKey Property [page 351]

1.1.21.14 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 unsafe string ReferencedTableName {get;}
```

Remarks

A string specifying the name of the referenced primary table.

Related Information

IsForeignKey Property [page 351]

1.1.22 ULInfoMessageEventArgs Class

Provides data for the ULConnection.InfoMessage event.

Syntax

Visual Basic

```
Public NotInheritable Class ULInfoMessageEventArgs Inherits
System.EventArgs
```

C#

```
public sealed class ULInfoMessageEventArgs : System.EventArgs
```

Members

All members of ULInfoMessageEventArgs, including inherited members.

Methods

Modifier and Type	Method	Description
public override string	ToString() [page 357]	The string representation of the ULConnection.InfoMessage event.

Properties

Modifier and Type	Property	Description
public string	Message [page 358]	The informational or warning message string returned by the database.
public ULSQLCode	NativeError [page 358]	The SQLCODE corresponding to the informational message or warning returned by the database.
public string	Source [page 359]	The name of the ADO.NET data provider returning the message.

In this section:

ToString() Method [page 357]
The string representation of the ULConnection.InfoMessage event.

Message Property [page 358]

The informational or warning message string returned by the database.

NativeError Property [page 358]
The SQLCODE corresponding to the informational message or warning returned by the database.

Source Property [page 359]
The name of the ADO.NET data provider returning the message.

Related Information

InfoMessage Event [page 170]

1.1.22.1 ToString() Method

The string representation of the ULConnection.InfoMessage event.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

Returns

The informational or warning message string.

Remarks

A string representation of the ULConnection.InfoMessage event.

Related Information

InfoMessage Event [page 170]

1.1.22.2 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;}
```

Remarks

A string containing the informational or warning message.

1.1.22.3 NativeError Property

The SQLCODE 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;}
```

Remarks

An informational or warning ULSQLCode value.

1.1.22.4 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;}
```

Remarks

The string "UltraLite.NET Data Provider".

1.1.23 ULMetaDataCollectionNames Class

Provides a list of constants for use with the ULConnection.GetSchema(String,String[]) method to retrieve metadata collections.

Syntax

Visual Basic

```
Public NotInheritable Class ULMetaDataCollectionNames
```

C#

```
public sealed class ULMetaDataCollectionNames
```

Members

All members of ULMetaDataCollectionNames, including inherited members.

Properties

Modifier and Type	Property	Description
public string	Columns [page 361]	Provides a constant for use with the UL-Connection.GetSchema(String) method that represents the Columns collection.

Modifier and Type	Property	Description
public string	DataSourceInformation [page 362]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataSourceInformation collection.
public string	DataTypes [page 363]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataTypes collection.
public string	ForeignKeys [page 364]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ForeignKeys collection.
public string	IndexColumns [page 365]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the IndexColumns collection.
public string	Indexes [page 366]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Indexes collection.
public string	MetaDataCollections [page 367]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the MetaDataCollections collection.
public string	Publications [page 368]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Publications collection.
public string	ReservedWords [page 369]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ReservedWords collection.
public string	Restrictions [page 370]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Restrictions collection.
public string	Tables [page 371]	Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Tables collection.

In this section:

Columns Property [page 361]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Columns collection.

DataSourceInformation Property [page 362]

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataSourceInformation collection.

DataTypes Property [page 363]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataTypes collection.

ForeignKeys Property [page 364]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ForeignKeys collection.

IndexColumns Property [page 365]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the IndexColumns collection.

Indexes Property [page 366]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Indexes collection.

MetaDataCollections Property [page 367]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the MetaDataCollections collection.

Publications Property [page 368]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Publications collection.

ReservedWords Property [page 369]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ReservedWords collection.

Restrictions Property [page 370]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Restrictions collection.

Tables Property [page 371]
Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Tables collection.

1.1.23.1 Columns Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Columns collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property Columns As String
```

C#

```
public string Columns {get;}
```

Remarks

A string representing the name of the Columns collection.

Example

The following code fills a DataTable object with the Columns collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.Columns )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.Columns );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.2 DataSourceInformation Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataSourceInformation collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property DataSourceInformation As String
```

C#

```
public string DataSourceInformation {get;}
```

Remarks

A string representing the name of the DataSourceInformation collection.

Example

The following code fills a DataTable object with the DataSourceInformation collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.DataSourceInformation )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.DataSourceInformation );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.3 DataTypes Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the DataTypes collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property DataTypes As String
```

C#

```
public string DataTypes {get;}
```

Remarks

A string representing the name of the DataTypes collection.

Example

The following code fills a DataTable object with the DataTypes collection.

```
' Visual Basic
```

```
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.DataTypes )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.DataTypes );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.4 ForeignKeys Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ForeignKeys collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property ForeignKeys As String
```

C#

```
public string ForeignKeys {get;}
```

Remarks

A string representing the name of the ForeignKeys collection.

Example

The following code fills a DataTable object with the ForeignKeys collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.ForeignKeys )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
```

```
   conn.GetSchema( ULMetaDataCollectionNames.ForeignKeys );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.5 IndexColumns Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the IndexColumns collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property IndexColumns As String
```

C#

```
public string IndexColumns {get;}
```

Remarks

A string representing the name of the IndexColumns collection.

Example

The following code fills a DataTable object with the IndexColumns collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.IndexColumns )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.IndexColumns );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.6 Indexes Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Indexes collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property Indexes As String
```

C#

```
public string Indexes {get;}
```

Remarks

A string representing the name of the Indexes collection.

Example

The following code fills a DataTable object with the Indexes collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.Indexes )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.Indexes );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.7 MetaDataCollections Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the MetaDataCollections collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property MetaDataCollections As String
```

C#

```
public string MetaDataCollections {get;}
```

Remarks

A string representing the name of the MetaDataCollections collection.

Example

The following code fills a DataTable object with the MetaDataCollections collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.MetaDataCollections )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.MetaDataCollections );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.8 Publications Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Publications collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property Publications As String
```

C#

```
public string Publications {get;}
```

Remarks

A string representing the name of the Publications collection.

Example

The following code fills a DataTable object with the Publications collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.Publications )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.Publications );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.9 ReservedWords Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the ReservedWords collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property ReservedWords As String
```

C#

```
public string ReservedWords {get;}
```

Remarks

A string representing the name of the ReservedWords collection.

Example

The following code fills a DataTable object with the ReservedWords collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.ReservedWords )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.ReservedWords );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.10 Restrictions Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Restrictions collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property Restrictions As String
```

C#

```
public string Restrictions {get;}
```

Remarks

A string representing the name of the Restrictions collection.

Example

The following code fills a DataTable object with the Restrictions collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.Restrictions )
```

The following code is the C# language equivalent:

```
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.Restrictions );
```

Related Information

GetSchema(string) Method [page 146]

1.1.23.11 Tables Property

Provides a constant for use with the ULConnection.GetSchema(String) method that represents the Tables collection.

Syntax

Visual Basic

```
Public Shared ReadOnly Property Tables As String
```

C#

```
public string Tables {get;}
```

Remarks

A string representing the name of the Tables collection.

Example

The following code fills a DataTable object with the Tables collection.

```
' Visual Basic
Dim schema As DataTable = _
   conn.GetSchema( ULMetaDataCollectionNames.Tables )
// C#
DataTable schema =
   conn.GetSchema( ULMetaDataCollectionNames.Tables );
```

Related Information

GetSchema(string) Method [page 146]

1.1.24 ULParameter Class

Represents a parameter to a ULCommand object.

Syntax

Visual Basic

```
Public NotInheritable Class ULParameter Inherits
System.Data.Common.DbParameter Implements System.ICloneable
```

C#

```
public sealed class ULParameter : System.Data.Common.DbParameter,
System.ICloneable
```

Members

All members of ULParameter, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULParameter [page 375]	Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

Methods

Modifier and Type	Method	Description
public override void	ResetDbType() [page 382]	This method is not supported in UltraLite.NET.
public override string	ToString() [page 383]	Returns the string representation of this instance.

Properties

Modifier and Type	Property	Description
public override DbType	DbType [page 383]	Specifies the System.Data.DbType for the parameter.
public override ParameterDirection	Direction [page 384]	A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.
public override bool	IsNullable [page 385]	Specifies whether the parameter accepts null values.
public int	Offset [page 385]	Specifies the offset to the ULParameter.Value.

Modifier and Type	Property	Description
public override string	ParameterName [page 386]	Specifies the name of the parameter.
public byte	Precision [page 387]	Specifies the maximum number of digits used to represent the ULParameter.Value property.
public byte	Scale [page 387]	Specifies the number of decimal places to which ULParameter.Value property is resolved.
public override int	Size [page 388]	Specifies the maximum size of the data within the column.
public override string	SourceColumn [page 389]	Specifies the name of the source column mapped to the DataSet object and used for loading or returning the value.
public override bool	SourceColumnNullMapping [page 389]	Specifies whether the source column is nullable.
public override DataRowVersion	SourceVersion [page 390]	The System.Data.DataRowVersion value to use when loading the ULParameter.Value property.
public ULDbType	ULDbType [page 390]	Specifies the Sap.Data.UltraLite.ULDbType for the parameter.
public override object	Value [page 391]	Specifies the value of the parameter.

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 the object type when using the ULParameter(string,object) constructor. For example:

```
' Visual Basic
Dim p As ULParameter = New ULParameter( "", CType( 0, Object ) )
```

The following code is the C# language equivalent:

```
// C#
ULParameter p = new ULParameter( "", (object)0 );
```

Parameters (including those created by the ULCommand.CreateParameter method) 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 property is important.

In this section:

ULParameter Constructor [page 375]

Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

ResetDbType() Method [page 382]
This method is not supported in UltraLite.NET.

ToString() Method [page 383]
Returns the string representation of this instance.

DbType Property [page 383]
Specifies the System.Data.DbType for the parameter.

Direction Property [page 384]
A value indicating whether the parameter is input-only, output-only, bidirectional, or a stored procedure return value parameter.

IsNullable Property [page 385]
Specifies whether the parameter accepts null values.

Offset Property [page 385]
Specifies the offset to the ULParameter.Value.

ParameterName Property [page 386]
Specifies the name of the parameter.

Precision Property [page 387]
Specifies the maximum number of digits used to represent the ULParameter.Value property.

Scale Property [page 387]
Specifies the number of decimal places to which ULParameter.Value property is resolved.

Size Property [page 388]
Specifies the maximum size of the data within the column.

SourceColumn Property [page 389]
Specifies the name of the source column mapped to the DataSet object and used for loading or returning the value.

SourceColumnNullMapping Property [page 389]
Specifies whether the source column is nullable.

SourceVersion Property [page 390]
The System.Data.DataRowVersion value to use when loading the ULParameter.Value property.

ULDbType Property [page 390]
Specifies the Sap.Data.UltraLite.ULDbType for the parameter.

Value Property [page 391]
Specifies the value of the parameter.

Related Information

ULCommand Class [page 45]
CreateParameter() Method [page 63]
ULParameter(string, object) Constructor [page 381]
Parameters Property [page 90]
Value Property [page 391]

1.1.24.1 ULParameter Constructor

Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

Overload List

Modifier and Type	Overload name	Description
public	ULParameter() [page 376]	Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.
public	ULParameter(string, ULDbType) [page 376]	Initializes a ULParameter object with the specified parameter name and data type.
public	ULParameter(string, ULDbType, int) [page 377]	Initializes a ULParameter object with the specified parameter name and data type.
public	ULParameter(string, ULDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object) [page 379]	Initializes a ULParameter object with the specified parameter name, data type, length, direction, nullability, numeric precision, numeric scale, source column, source version, and value.
public	ULParameter(string, ULDbType, int, string) [page 380]	Initializes a ULParameter object with the specified parameter name, data type, and length.
public	ULParameter(string, object) [page 381]	Initializes a ULParameter object with the specified parameter name and value.

In this section:

ULParameter() Constructor [page 376]
Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

ULParameter(string, ULDbType) Constructor [page 376]
Initializes a ULParameter object with the specified parameter name and data type.

ULParameter(string, ULDbType, int) Constructor [page 377]
Initializes a ULParameter object with the specified parameter name and data type.

ULParameter(string, ULDbType, int, ParameterDirection, bool, byte, byte, string, DataRowVersion, object) Constructor [page 379]
Initializes a ULParameter object with the specified parameter name, data type, length, direction, nullability, numeric precision, numeric scale, source column, source version, and value.

ULParameter(string, ULDbType, int, string) Constructor [page 380]
Initializes a ULParameter object with the specified parameter name, data type, and length.

ULParameter(string, object) Constructor [page 381]
Initializes a ULParameter object with the specified parameter name and value.

1.1.24.1.1 ULParameter() Constructor

Initializes a ULParameter object with null (Nothing in Visual Basic) as its value.

Syntax

Visual Basic

```
Public Sub ULParameter ()
```

C#

```
public ULParameter ()
```

Example

The following code creates a ULParameter value of 3 and adds it to a ULCommand object named cmd.

```
' Visual Basic
Dim p As ULParameter = New ULParameter
p.Value = 3
cmd.Parameters.Add( p )
```

The following code is the C# language equivalent:

```
// C#
ULParameter p = new ULParameter();
p.Value = 3;
cmd.Parameters.Add( p );
```

Related Information

Value Property [page 391]
ULParameter(string, object) Constructor [page 381]
ULCommand Class [page 45]

1.1.24.1.2 ULParameter(string, ULDbType) Constructor

Initializes a ULParameter object with the specified parameter name and data type.

Syntax

Visual Basic

```
Public Sub ULParameter (
    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 the ULCommand object.

dbType One of the Sap.Data.UltraLite.ULDbType values.

Remarks

This constructor is not recommended; it is provided for compatibility with other data providers.

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

ULParameter() Constructor [page 376]
ULParameter(string, object) Constructor [page 381]
Value Property [page 391]
ULCommand Class [page 45]

1.1.24.1.3 ULParameter(string, ULDbType, int) Constructor

Initializes a ULParameter object with the specified parameter name and data type.

Syntax

Visual Basic

```
Public Sub ULParameter (
    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 the ULCommand object.

dbType One of the Sap.Data.UltraLite.ULDbType values.

size The length of the parameter.

Remarks

This constructor is not recommended; it is provided for compatibility with other data providers.

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

ULParameter() Constructor [page 376]
ULParameter(string, object) Constructor [page 381]
Value Property [page 391]

1.1.24.1.4 ULParameter(string, ULDbType, int, ParameterDirection, bool, 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.

Syntax

Visual Basic

```
Public Sub ULParameter (
    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 the ULCommand object.

dbType One of the Sap.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 the Value property is resolved.

scale The total number of decimal places to which the Value property is resolved.

sourceColumn The name of the source column to map.

sourceVersion One of the System.Data.DataRowVersion values.
value Pass a System.Object to produce the value of the parameter.

Exceptions

ULException class Only the System.Data.ParameterDirection.Input direction is supported in UltraLite.NET.

Remarks

This constructor is not recommended; it is provided for compatibility with other data providers.

Related Information

ULParameter() Constructor [page 376]
ULParameter(string, object) Constructor [page 381]
ULCommand Class [page 45]

1.1.24.1.5 ULParameter(string, ULDbType, int, string) Constructor

Initializes a ULParameter object with the specified parameter name, data type, and length.

Syntax

Visual Basic

```
Public Sub ULParameter (
    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 the ULCommand object.

dbType One of the Sap.Data.UltraLite.ULDbType values.

size The length of the parameter.

sourceColumn The name of the source column to map.

Remarks

This constructor is not recommended; it is provided for compatibility with other data providers.

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

ULParameter() Constructor [page 376]
ULParameter(string, object) Constructor [page 381]
Value Property [page 391]
ULCommand Class [page 45]

1.1.24.1.6 ULParameter(string, object) Constructor

Initializes a ULParameter object with the specified parameter name and value.

Syntax

Visual Basic

```
Public Sub ULParameter (
    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 the ULCommand object.

value Pass a System.Object class to produce 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 the object type when using this constructor.

Example

The following code creates a ULParameter value of 0 and adds it to a ULCommand object named cmd.

```
' Visual Basic
cmd.Parameters.Add( New ULParameter( "", CType( 0, Object ) ) )
```

The following code is the C# language equivalent:

```
// C#
cmd.Parameters.Add( new ULParameter( "", (object)0 ) );
```

Related Information

ULParameter() Constructor [page 376]
ULCommand Class [page 45]

1.1.24.2 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 property is important.

Related Information

Value Property [page 391]

1.1.24.3 ToString() Method

Returns the string representation of this instance.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

Returns

The name of the parameter.

1.1.24.4 DbType Property

Specifies the System.Data.DbType for the parameter.

Syntax

Visual Basic

```
Public Overrides Property DbType As DbType
```

C#

```
public override DbType DbType {get;set;}
```

Remarks

One of the System.Data.DbType values.

The ULParameter.ULDbType and DbType properties are linked. Therefore, setting the DbType property changes the ULParameter.ULDbType property to a supporting Sap.Data.UltraLite.ULDbType value.

Related Information

ULDbType Property [page 390]

1.1.24.5 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;}
```

Remarks

One of the System.Data.ParameterDirection values.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.6 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;}
```

Remarks

True if null values are accepted, false otherwise. The default is false. Null values are handled using the DBNull class.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.7 Offset Property

Specifies the offset to the ULParameter.Value.

Syntax

Visual Basic

```
Public Property Offset As Integer
```

C#

```
public int Offset {get;set;}
```

Remarks

The offset to the value. The default is 0.

In UltraLite.NET parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.8 ParameterName Property

Specifies the name of the parameter.

Syntax

Visual Basic

```
Public Overrides Property ParameterName As String
```

C#

```
public override string ParameterName {get;set;}
```

Remarks

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.

In UltraLite.NET, parameter names are not used by an ULCommand object. All parameters are treated as positional parameters and are used by a command in the order that they were added.

Related Information

ULCommand Class [page 45]

1.1.24.9 Precision Property

Specifies the maximum number of digits used to represent the ULParameter.Value property.

Syntax

Visual Basic

```
Public Property Precision As Byte
```

C#

```
public byte Precision {get;set;}
```

Remarks

The maximum number of digits used to represent the ULParameter.Value property. The default value is 0, which indicates that the data provider sets the precision for the ULParameter.Value property.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.10 Scale Property

Specifies the number of decimal places to which ULParameter.Value property is resolved.

Syntax

Visual Basic

```
Public Property Scale As Byte
```

C#

```
public byte Scale {get;set;}
```

Remarks

The number of decimal places to which ULParameter.Value property is resolved. The default is 0.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.11 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;}
```

Remarks

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.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.12 SourceColumn Property

Specifies the name of the source column mapped to the DataSet object and used for loading or returning the value.

Syntax

Visual Basic

```
Public Overrides Property SourceColumn As String
```

C#

```
public override string SourceColumn {get;set;}
```

Remarks

A string specifying the name of the source column mapped to the DataSet object and used for loading or returning the value.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.13 SourceColumnNullMapping Property

Specifies whether the source column is nullable.

Syntax

Visual Basic

```
Public Overrides Property SourceColumnNullMapping As Boolean
```

C#

```
public override bool SourceColumnNullMapping {get;set;}
```

Remarks

True if the source column is nullable; false, otherwise.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]

1.1.24.14 SourceVersion Property

The System.Data.DataRowVersion value to use when loading the ULParameter.Value property.

Syntax

Visual Basic

```
Public Overrides Property SourceVersion As DataRowVersion
```

C#

```
public override DataRowVersion SourceVersion {get;set;}
```

Related Information

Value Property [page 391]

1.1.24.15 ULDbType Property

Specifies the Sap.Data.UltraLite.ULDbType for the parameter.

Syntax

Visual Basic

```
Public Property ULDbType As ULDbType
```

C#

```
public ULDbType ULDbType {get;set;}
```

Remarks

One of the Sap.Data.UltraLite.ULDbType values.

The ULDbType and ULParameter.DbType properties are linked. Therefore, setting the ULDbType property changes the ULParameter.DbType property to a supporting System.Data.DbType value.

In UltraLite.NET, parameters can only be used as IN parameters and all mapping information is ignored. Only the ULParameter.Value property is important.

Related Information

Value Property [page 391]
DbType Property [page 383]

1.1.24.16 Value Property

Specifies the value of the parameter.

Syntax

Visual Basic

```
Public Overrides Property Value As Object
```

C#

```
public override object Value {get;set;}
```

Remarks

Pass a System.Object class that specifies the value of the parameter.

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 object with ULSQLCode.SQLE_CONVERSION_ERROR if it cannot convert the value.

Related Information

ULException Class [page 312]

1.1.25 ULParameterCollection Class

Represents all parameters to a ULCommand object.

Syntax

Visual Basic

```
Public NotInheritable Class ULParameterCollection Inherits
System.Data.Common.DbParameterCollection
```

C#

```
public sealed class ULParameterCollection :
System.Data.Common.DbParameterCollection
```

Members

All members of ULParameterCollection, including inherited members.

Methods

Modifier and Type	Method	Description
public ULParameter	Add [page 395]	Adds a ULParameter object to the collection.
public override void	AddRange [page 403]	Adds an array of values to the end of the ULParameterCollection.
public override void	Clear() [page 405]	Removes all the parameters from the collection.
public override bool	Contains [page 405]	Checks whether a ULParameter object exists in the collection.
public override void	CopyTo(Array, int) [page 407]	Copies ULParameter objects from the ULParameterCollection to the specified array.
public override IEnumerator	GetEnumerator() [page 408]	Returns an enumerator for the collection.
protected override DbParameter	GetParameter [page 409]	See individual topics
public override int	IndexOf [page 410]	Returns the location of the ULParameter object in the collection.
public override void	Insert(int, object) [page 412]	Inserts an ULParameter object in the collection at the specified index.
public override void	Remove(object) [page 413]	Removes an ULParameter object from the collection.
public override void	RemoveAt [page 414]	Removes the parameter at the specified index in the collection.

Modifier and Type	Method	Description
protected override void	SetParameter [page 416]	See individual topics

Properties

Modifier and Type	Property	Description
public override int	Count [page 417]	Returns the number of ULParameter objects in the collection.
public override bool	IsFixedSize [page 417]	Indicates whether the ULParameterCollection object has a fixed size.
public override bool	IsReadOnly [page 418]	Indicates whether the ULParameterCollection object is read-only.
public override bool	IsSynchronized [page 418]	Indicates whether the ULParameterCollection object is synchronized.
public override object	SyncRoot [page 419]	Returns an object that can be used to synchronize access to the SAParameterCollection object.
public new ULParameter	this [page 419]	Returns the ULParameter object at the specified index.

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

There is no constructor for the ULParameterCollection class. You obtain a ULParameterCollection object from the ULCommand.Parameters property.

In this section:

Add Method [page 395]
: Adds a ULParameter object to the collection.

AddRange Method [page 403]
: Adds an array of values to the end of the ULParameterCollection.

Clear() Method [page 405]
: Removes all the parameters from the collection.

Contains Method [page 405]
: Checks whether a ULParameter object exists in the collection.

CopyTo(Array, int) Method [page 407]
: Copies ULParameter objects from the ULParameterCollection to the specified array.

GetEnumerator() Method [page 408]
Returns an enumerator for the collection.

GetParameter Method [page 409]

IndexOf Method [page 410]
Returns the location of the ULParameter object in the collection.

Insert(int, object) Method [page 412]
Inserts an ULParameter object in the collection at the specified index.

Remove(object) Method [page 413]
Removes an ULParameter object from the collection.

RemoveAt Method [page 414]
Removes the parameter at the specified index in the collection.

SetParameter Method [page 416]

Count Property [page 417]
Returns the number of ULParameter objects in the collection.

IsFixedSize Property [page 417]
Indicates whether the ULParameterCollection object has a fixed size.

IsReadOnly Property [page 418]
Indicates whether the ULParameterCollection object is read-only.

IsSynchronized Property [page 418]
Indicates whether the ULParameterCollection object is synchronized.

SyncRoot Property [page 419]
Returns an object that can be used to synchronize access to the SAParameterCollection object.

this Property [page 419]
Returns the ULParameter object at the specified index.

Related Information

ULCommand Class [page 45]
CommandText Property [page 85]
Parameters Property [page 90]

1.1.25.1 Add Method

Adds a ULParameter object to the collection.

Overload List

Modifier and Type	Overload name	Description
public ULParameter	Add(ULParameter) [page 396]	Adds a ULParameter object to the collection.
public override int	Add(object) [page 397]	Adds a ULParameter object to the collection.
public ULParameter	Add(string, ULDbType) [page 398]	Adds a new ULParameter object, created using the specified parameter name and data type, to the collection.
public ULParameter	Add(string, ULDbType, int) [page 399]	Adds a new ULParameter object, created using the specified parameter name, data type, and length, to the collection.
public ULParameter	Add(string, ULDbType, int, string) [page 400]	Adds a new ULParameter object, created using the specified parameter name, data type, length, and source column name, to the collection.
public ULParameter	Add(string, object) [page 402]	Adds a new ULParameter object, created using the specified parameter name and value, to the collection.

In this section:

Add(ULParameter) Method [page 396]
Adds a ULParameter object to the collection.

Add(object) Method [page 397]
Adds a ULParameter object to the collection.

Add(string, ULDbType) Method [page 398]
Adds a new ULParameter object, created using the specified parameter name and data type, to the collection.

Add(string, ULDbType, int) Method [page 399]
Adds a new ULParameter object, created using the specified parameter name, data type, and length, to the collection.

Add(string, ULDbType, int, string) Method [page 400]
Adds a new ULParameter object, created using the specified parameter name, data type, length, and source column name, to the collection.

Add(string, object) Method [page 402]
Adds a new ULParameter object, created using the specified parameter name and value, to the collection.

1.1.25.1.1 Add(ULParameter) Method

Adds a ULParameter object 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.

Returns

The new ULParameter object.

Exceptions

ArgumentNullException The value cannot be null (Nothing in Visual Basic).
ArgumentException The ULParameter object can only be added to the collection once.

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

Related Information

Add(string, object) Method [page 402]
ULParameter Class [page 372]
CommandText Property [page 85]

1.1.25.1.2 Add(object) Method

Adds a ULParameter object 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.

Returns

The index of the new ULParameter object.

Exceptions

ArgumentNullException The value cannot be null (Nothing in Visual Basic).
InvalidCastException The value specified must be a ULParameter object.
ArgumentException The ULParameter object can only be added to the collection once.

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

Related Information

Add(ULParameter) Method [page 396]
Add(string, object) Method [page 402]
CommandText Property [page 85]
ULParameter Class [page 372]

1.1.25.1.3 Add(string, ULDbType) Method

Adds a new ULParameter object, 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 the ULCommand object.

ulDbType One of the Sap.Data.UltraLite.ULDbType values.

Returns

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

Related Information

Add(ULParameter) Method [page 396]
Add(string, object) Method [page 402]
ULParameter Class [page 372]
CommandText Property [page 85]
ULCommand Class [page 45]

1.1.25.1.4 Add(string, ULDbType, int) Method

Adds a new ULParameter object, 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 the ULCommand object.

ulDbType One of the Sap.Data.UltraLite.ULDbType values.
size The length of the parameter.

Returns

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

Related Information

Add(ULParameter) Method [page 396]
Add(string, object) Method [page 402]
ULParameter Class [page 372]
CommandText Property [page 85]
ULCommand Class [page 45]

1.1.25.1.5 Add(string, ULDbType, int, string) Method

Adds a new ULParameter object, 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 the ULCommand object.

ulDbType One of the Sap.Data.UltraLite.ULDbType values.

size The length of the parameter.

sourceColumn The name of the source column to map.

Returns

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 value. 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 value as there are parameters in the collection. Nulls are substituted for missing parameters.

Related Information

Add(ULParameter) Method [page 396]
Add(string, object) Method [page 402]
ULParameter Class [page 372]
CommandText Property [page 85]
ULCommand Class [page 45]

1.1.25.1.6 Add(string, object) Method

Adds a new ULParameter object, 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 the ULCommand object.

value A System.Object that is to be the value of the parameter.

Returns

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 value. 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 value 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 the object type when using this method.

Example

The following code adds a ULParameter value of 0 to a ULCommand object named cmd.

```
' Visual Basic
cmd.Parameters.Add( "", CType( 0, Object ) )
```

The following code is the C# language equivalent:

```
// C#
cmd.Parameters.Add( "", (object)0 );
```

Related Information

Add(ULParameter) Method [page 396]
ULParameter Class [page 372]
CommandText Property [page 85]
ULCommand Class [page 45]

1.1.25.2 AddRange Method

Adds an array of values to the end of the ULParameterCollection.

Overload List

Modifier and Type	Overload name	Description
public override void	AddRange(Array) [page 404]	Adds an array of values to the end of the ULParameterCollection.
public void	AddRange(ULParameter[]) [page 404]	Adds an array of values to the end of the ULParameterCollection.

In this section:

AddRange(Array) Method [page 404]
Adds an array of values to the end of the ULParameterCollection.

AddRange(ULParameter[]) Method [page 404]
Adds an array of values to the end of the ULParameterCollection.

1.1.25.2.1 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.

Related Information

ULParameter Class [page 372]

1.1.25.2.2 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 the DbParameterCollection.AddRange(Array) method.

Related Information

ULParameter Class [page 372]
ULParameterCollection Class [page 392]

1.1.25.3 Clear() Method

Removes all the parameters from the collection.

Syntax

Visual Basic

```
Public Overrides Sub Clear ()
```

C#

```
public override void Clear ()
```

1.1.25.4 Contains Method

Checks whether a ULParameter object exists in the collection.

Overload List

Modifier and Type	Overload name	Description
public override bool	Contains(object) [page 406]	Checks whether a ULParameter object exists in the collection.
public override bool	Contains(string) [page 407]	Checks whether a ULParameter object with the specified name exists in the collection.

In this section:

Contains(object) Method [page 406]

Checks whether a ULParameter object exists in the collection.

Contains(string) Method [page 407]
Checks whether a ULParameter object with the specified name exists in the collection.

1.1.25.4.1 Contains(object) Method

Checks whether a ULParameter object 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.

Returns

True if the collection contains the ULParameter object, false otherwise.

Related Information

Contains(string) Method [page 407]
ULParameter Class [page 372]

1.1.25.4.2 Contains(string) Method

Checks whether a ULParameter object 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.

Returns

True if the collection contains the ULParameter object, false otherwise.

Related Information

Contains(object) Method [page 406]
ULParameter Class [page 372]

1.1.25.5 CopyTo(Array, int) 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.

Related Information

ULParameter Class [page 372]

1.1.25.6 GetEnumerator() Method

Returns an enumerator for the collection.

Syntax

Visual Basic

```
Public Overrides Function GetEnumerator () As
System.Collections.IEnumerator
```

C#

```
public override IEnumerator GetEnumerator ()
```

Returns

An ArrayList enumerator enumerating the parameters in the collection.

1.1.25.7 GetParameter Method

Overload List

Modifier and Type	Overload name	Description
protected override DbParameter	GetParameter(int) [page 409]	
protected override DbParameter	GetParameter(string) [page 409]	

In this section:

1.1.25.7.1 GetParameter(int) Method

Syntax

Visual Basic

```
Protected Overrides Function GetParameter (ByVal index As Integer) As
DbParameter
```

C#

```
protected override DbParameter GetParameter (int index)
```

1.1.25.7.2 GetParameter(string) Method

Syntax

Visual Basic

```
Protected Overrides Function GetParameter (ByVal parameterName As String)
As DbParameter
```

C#

```
protected override DbParameter GetParameter (string parameterName)
```

1.1.25.8 IndexOf Method

Returns the location of the ULParameter object in the collection.

Overload List

Modifier and Type	Overload name	Description
public override int	IndexOf(object) [page 410]	Returns the location of the ULParameter object in the collection.
public override int	IndexOf(string) [page 411]	Returns the location of the ULParameter object with the specified name in the collection.

In this section:

IndexOf(object) Method [page 410]
Returns the location of the ULParameter object in the collection.

IndexOf(string) Method [page 411]
Returns the location of the ULParameter object with the specified name in the collection.

1.1.25.8.1 IndexOf(object) Method

Returns the location of the ULParameter object 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.

Returns

The zero-based index of the ULParameter object in the collection or -1 if the parameter is not found.

Exceptions

InvalidCastException The value specified must be a ULParameter object.

Related Information

IndexOf(string) Method [page 411]
ULParameter Class [page 372]

1.1.25.8.2 IndexOf(string) Method

Returns the location of the ULParameter object 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.

Returns

The zero-based index of the ULParameter object in the collection or -1 if the parameter is not found.

Related Information

IndexOf(object) Method [page 410]
ULParameter Class [page 372]

1.1.25.9 Insert(int, object) Method

Inserts an ULParameter object 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.

Exceptions

IndexOutOfRangeException The index is invalid.
ArgumentNullException You cannot set a parameter using a null reference (Nothing in Visual Basic).
InvalidCastException The value specified must be a ULParameter object.

Related Information

ULParameter Class [page 372]

1.1.25.10 Remove(object) Method

Removes an ULParameter object 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.

Exceptions

ArgumentNullException You cannot set a parameter using a null reference (Nothing in Visual Basic).
InvalidCastException The value specified must be a ULParameter object.
ArgumentException The collection does not contain the specified parameter.

Related Information

ULParameter Class [page 372]

1.1.25.11 RemoveAt Method

Removes the parameter at the specified index in the collection.

Overload List

Modifier and Type	Overload name	Description
public override void	RemoveAt(int) [page 414]	Removes the parameter at the specified index in the collection.
public override void	RemoveAt(string) [page 415]	Removes the parameter with the specified name from the collection.

In this section:

RemoveAt(int) Method [page 414]
Removes the parameter at the specified index in the collection.

RemoveAt(string) Method [page 415]
Removes the parameter with the specified name from the collection.

1.1.25.11.1 RemoveAt(int) 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.

Exceptions

IndexOutOfRangeException The index is invalid.

Related Information

RemoveAt(string) Method [page 415]
Count Property [page 417]

1.1.25.11.2 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.

Exceptions

IndexOutOfRangeException There is no parameter with the specified name.

Related Information

RemoveAt(int) Method [page 414]

1.1.25.12 SetParameter Method

Overload List

Modifier and Type	Overload name	Description
protected override void	SetParameter(int, DbParameter) [page 416]	
protected override void	SetParameter(string, DbParameter) [page 416]	

In this section:

1.1.25.12.1 SetParameter(int, DbParameter) Method

Syntax

Visual Basic

```
Protected Overrides Sub SetParameter (
    ByVal index As Integer,
    ByVal parm As DbParameter
)
```

C#

```
protected override void SetParameter (
    int index,
    DbParameter parm
)
```

1.1.25.12.2 SetParameter(string, DbParameter) Method

Syntax

Visual Basic

```
Protected Overrides Sub SetParameter (
    ByVal parameterName As String,
```

```
    ByVal parm As DbParameter
)
```

C#

```
protected override void SetParameter (
    string parameterName,
    DbParameter parm
)
```

1.1.25.13 Count Property

Returns the number of ULParameter objects in the collection.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Count As Integer
```

C#

```
public override int Count {get;}
```

Remarks

The number of ULParameter objects in the collection.

1.1.25.14 IsFixedSize Property

Indicates whether the ULParameterCollection object has a fixed size.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property IsFixedSize As Boolean
```

C#

```
public override bool IsFixedSize {get;}
```

Remarks

True if this collection has a fixed size, false otherwise.

1.1.25.15 IsReadOnly Property

Indicates whether the ULParameterCollection object is read-only.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property IsReadOnly As Boolean
```

C#

```
public override bool IsReadOnly {get;}
```

Remarks

True if this collection is read-only, false otherwise.

1.1.25.16 IsSynchronized Property

Indicates whether the ULParameterCollection object is synchronized.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property IsSynchronized As Boolean
```

C#

```
public override bool IsSynchronized {get;}
```

Remarks

True if this collection is synchronized, false otherwise.

1.1.25.17 SyncRoot Property

Returns an object that can be used to synchronize access to the SAParameterCollection object.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property SyncRoot As Object
```

C#

```
public override object SyncRoot {get;}
```

Remarks

The object to be used to synchronize access to this collection.

1.1.25.18 this Property

Returns the ULParameter object at the specified index.

Overload List

Modifier and Type	Overload name	Description
public new ULParameter	this[int index] [page 420]	Returns the ULParameter object at the specified index.
public new ULParameter	this[string parameterName] [page 420]	Returns the ULParameter object with the specified name.

In this section:

this[int index] Property [page 420]
Returns the ULParameter object at the specified index.

this[string parameterName] Property [page 420]
Returns the ULParameter object with the specified name.

1.1.25.18.1 this[int index] Property

Returns the ULParameter object at the specified index.

Syntax

Visual Basic

```
Public Shadows Property Item (ByVal index As Integer) As ULParameter
```

C#

```
public new ULParameter this[int index] {get;set;}
```

Returns

The ULParameter object at the specified index.

Remarks

In C#, this property is the indexer for the ULParameterCollection class.

This is the strongly-typed version of DbParameterCollection.this[int] property.

Related Information

ULParameter Class [page 372]

1.1.25.18.2 this[string parameterName] Property

Returns the ULParameter object with the specified name.

Syntax

Visual Basic

```
Public Shadows Property Item (ByVal parameterName As String) As ULParameter
```

C#

```
public new ULParameter this[string parameterName] {get;set;}
```

Returns

The ULParameter object with the specified name.

Remarks

In C#, this property is the indexer for the ULParameterCollection class.

This is the strongly-typed version of DbParameterCollection.this[string] property.

Related Information

GetOrdinal(string) Method [page 286]
GetValue(int) Method [page 295]
GetFieldType(int) Method [page 279]
ULParameter Class [page 372]

1.1.26 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
```

Members

All members of ULResultSet, including inherited members.

Methods

Modifier and Type	Method	Description
public unsafe void	AppendBytes(int, byte[], int, int) [page 428]	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDb-Type.LongBinary column.
public unsafe void	AppendChars(int, char[], int, int) [page 429]	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDb-Type.LongVarchar column.
public void	Delete() [page 431]	Deletes the current row.
public void	SetBoolean(int, bool) [page 431]	Sets the value for the specified column using a System.Boolean.
public void	SetByte(int, byte) [page 432]	Sets the value for the specified column using a System.Byte (unsigned 8-bit in-teger).
public unsafe void	SetBytes(int, byte[]) [page 433]	Sets the value for the specified column using an array of System.Bytes.
public unsafe void	SetDateTime(int, DateTime) [page 435]	Sets the value for the specified column using a System.DateTime.
public void	SetDBNull(int) [page 436]	Sets a column to NULL.
public void	SetDecimal(int, decimal) [page 437]	Sets the value for the specified column using a System.Decimal.
public void	SetDouble(int, double) [page 438]	Sets the value for the specified column using a System.Double.
public void	SetFloat(int, float) [page 439]	Sets the value for the specified column using a System.Single.
public unsafe void	SetGuid(int, Guid) [page 440]	Sets the value for the specified column using a System.Guid.
public void	SetInt16(int, short) [page 441]	Sets the value for the specified column using a System.Int16.
public void	SetInt32(int, int) [page 442]	Sets the value for the specified column using a System.Int32.
public void	SetInt64(int, long) [page 443]	Sets the value for the specified column using an Int64.
public unsafe void	SetString(int, string) [page 445]	Sets the value for the specified column using a System.String.
public unsafe void	SetTimeSpan(int, TimeSpan) [page 446]	Sets the value for the specified column using a System.TimeSpan.
public void	SetToDefault(int) [page 447]	Sets the value for the specified column to its default value.
public void	SetUInt16(int, ushort) [page 448]	Sets the value for the specified column using a System.UInt16.
public void	SetUInt32(int, uint) [page 449]	Sets the value for the specified column using a System.UInt32.

Modifier and Type	Method	Description
public void	SetUInt64(int, ulong) [page 450]	Sets the value for the specified column using a System.UInt64.
public void	Update() [page 451]	Updates the current row with the current column values (specified using the set methods).
public void	UpdateBegin() [page 452]	Prepares to update the current row.

Inherited members from ULDataReader

Modifier and Type	Member	Description
public override void	Close() [page 266]	Closes the cursor.
public override int	Depth [page 305]	Returns the depth of nesting for the current row.
protected override void	Dispose(bool) [page 267]	
public override int	FieldCount [page 305]	Returns the number of columns in the cursor.
public override unsafe bool	GetBoolean(int) [page 267]	Returns the value for the specified column as a System.Boolean.
public override unsafe byte	GetByte(int) [page 268]	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
public unsafe byte[]	GetBytes(int) [page 269]	UL Ext: Returns the value for the specified column as an array of System.Bytes values.
public override unsafe long	GetBytes(int, long, byte[], int, int) [page 270]	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.
public override char	GetChar(int) [page 272]	This method is not supported in UltraLite.NET.
public override unsafe long	GetChars(int, long, char[], int, int) [page 273]	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.
public override string	GetDataTypeName(int) [page 274]	Returns the name of the specified column's provider data type.
public override unsafe DateTime	GetDateTime(int) [page 275]	Returns the value for the specified column as a System.DateTime type with millisecond accuracy.
protected override DbDataReader	GetDbDataReader(int) [page 276]	
public override decimal	GetDecimal(int) [page 276]	Returns the value for the specified column as a System.Decimal type.

Modifier and Type	Member	Description
public override unsafe double	GetDouble(int) [page 277]	Returns the value for the specified column as a System.Double type.
public override IEnumerator	GetEnumerator() [page 278]	Returns an System.Collections.IEnumerator value that iterates through the ULDataReader object.
public override Type	GetFieldType(int) [page 279]	Returns the System.Type value most appropriate for the specified column.
public override unsafe float	GetFloat(int) [page 280]	Returns the value for the specified column as a System.Single type.
public override unsafe Guid	GetGuid(int) [page 281]	Returns the value for the specified column as a UUID (System.Guid) type.
public override unsafe short	GetInt16(int) [page 282]	Returns the value for the specified column as a System.Int16 type.
public override unsafe int	GetInt32(int) [page 283]	Returns the value for the specified column as a System.Int32 type.
public override unsafe long	GetInt64(int) [page 284]	Returns the value for the specified column as a System.Int64 type.
public override string	GetName(int) [page 285]	Returns the name of the specified column.
public override unsafe int	GetOrdinal(string) [page 286]	Returns the column ID of the named column.
public unsafe int	GetRowCount(int) [page 287]	UL Ext: Returns the number of rows in the cursor, within threshold.
public override DataTable	GetSchemaTable() [page 288]	Returns a System.Data.DataTable value that describes the column metadata of the ULDataReader object.
public override unsafe String	GetString(int) [page 290]	Returns the value for the specified column as a System.String type.
public unsafe TimeSpan	GetTimeSpan(int) [page 291]	Returns the value for the specified column as a System.TimeSpan type with millisecond accuracy.
public unsafe ushort	GetUInt16(int) [page 292]	Returns the value for the specified column as a System.UInt16 type.
public unsafe uint	GetUInt32(int) [page 293]	Returns the value for the specified column as a System.UInt32 type.
public unsafe ulong	GetUInt64(int) [page 294]	Returns the value for the specified column as a System.UInt64 type.
public override object	GetValue(int) [page 295]	Returns the value of the specified column in its native format.
public override int	GetValues(object[]) [page 296]	Returns all the column values for the current row.
public override unsafe bool	HasRows [page 306]	Checks whether the ULDataReader object has one or more rows.

Modifier and Type	Member	Description
public unsafe bool	IsBOF [page 306]	UL Ext: Checks whether the current row position is before the first row.
public override bool	IsClosed [page 307]	Checks whether the cursor is currently open.
public override unsafe bool	IsDBNull(int) [page 297]	Checks whether the value from the specified column is NULL.
public unsafe bool	IsEOF [page 307]	UL Ext: Checks whether the current row position is after the last row.
public void	MoveAfterLast() [page 298]	UL Ext: Positions the cursor to after the last row of the cursor.
public void	MoveBeforeFirst() [page 298]	UL Ext: Positions the cursor to before the first row of the cursor.
public unsafe bool	MoveFirst() [page 299]	UL Ext: Positions the cursor to the first row of the cursor.
public unsafe bool	MoveLast() [page 299]	UL Ext: Positions the cursor to the last row of the cursor.
public unsafe bool	MoveNext() [page 300]	UL Ext: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
public unsafe bool	MovePrevious() [page 301]	UL Ext: Positions the cursor to the previous row or before the first row.
public unsafe bool	MoveRelative(int) [page 301]	UL Ext: Positions the cursor relative to the current row.
public override bool	NextResult() [page 302]	Advances the ULDataReader object to the next result when reading the results of batch SQL statements.
public override bool	Read() [page 303]	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
public override int	RecordsAffected [page 308]	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement.
public int	RowCount [page 308]	UL Ext: Returns the number of rows in the cursor.
public ULCursorSchema	Schema [page 309]	UL Ext: Holds the schema of this cursor.
public override object	this[int colID] [page 310]	Returns the value of the specified column in its native format.
public override object	this[string name] [page 311]	Returns the value of the specified named column in its native format.
protected void	Validate() [page 304]	
protected void	Validate(int) [page 304]	

Remarks

There is no constructor for this class. Result sets are created using the ULCommand.ExecuteResultSet method.

```
' Visual Basic
Dim cmd As ULCommand = new ULCommand( _
   "SELECT emp_id FROM employee", conn _
)
Dim resultSet As ULResultSet = cmd.ExecuteResultSet()
```

The following code is the C# language equivalent:

```
// 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 the ULCommand.ExecuteTable method or the ULDataAdapter class.

In this section:

SetGuid(int, Guid) Method [page 440]
Sets the value for the specified column using a System.Guid.

SetInt16(int, short) Method [page 441]
Sets the value for the specified column using a System.Int16.

SetInt32(int, int) Method [page 442]
Sets the value for the specified column using a System.Int32.

SetInt64(int, long) Method [page 443]
Sets the value for the specified column using an Int64.

SetString(int, string) Method [page 445]
Sets the value for the specified column using a System.String.

SetTimeSpan(int, TimeSpan) Method [page 446]
Sets the value for the specified column using a System.TimeSpan.

SetToDefault(int) Method [page 447]
Sets the value for the specified column to its default value.

SetUInt16(int, ushort) Method [page 448]
Sets the value for the specified column using a System.UInt16.

SetUInt32(int, uint) Method [page 449]
Sets the value for the specified column using a System.UInt32.

SetUInt64(int, ulong) Method [page 450]
Sets the value for the specified column using a System.UInt64.

Update() Method [page 451]
Updates the current row with the current column values (specified using the set methods).

UpdateBegin() Method [page 452]
Prepares to update the current row.

Related Information

ExecuteResultSet() Method [page 76]
ULCommand Class [page 45]
ExecuteTable() Method [page 81]
ULDataAdapter Class [page 224]
ULDataReader Class [page 260]

1.1.26.1 AppendBytes(int, byte[], int, int) 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 colID As Integer,
    ByVal val As Byte(),
    ByVal srcOffset As Integer,
    ByVal count As Integer
)
```

C#

```
public unsafe void AppendBytes (
    int colID,
    byte[] val,
    int srcOffset,
    int count
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

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 changes are not made permanent until 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 the *val* length.

For other errors, a ULException with the appropriate error code is thrown.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
InsertBegin() Method [page 528]
Insert() Method [page 527]
ULException Class [page 312]
ULException Class [page 312]
FieldCount Property [page 305]

1.1.26.2 AppendChars(int, char[], int, int) 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 colID As Integer,
    ByVal val As Char(),
    ByVal srcOffset As Integer,
    ByVal count As Integer
)
```

C#

```
public unsafe void AppendChars (
    int colID,
    char[] val,
    int srcOffset,
    int count
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

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 changes are not made permanent until 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 *val* length.

For other errors, a ULException with the appropriate error code is thrown.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
InsertBegin() Method [page 528]
Insert() Method [page 527]
ULException Class [page 312]
FieldCount Property [page 305]

1.1.26.3 Delete() Method

Deletes the current row.

Syntax

Visual Basic

```
Public Sub Delete ()
```

C#

```
public void Delete ()
```

Exceptions

ULException class A SQL error occurred.

Related Information

StartSynchronizationDelete() Method [page 155]
StopSynchronizationDelete() Method [page 156]

1.1.26.4 SetBoolean(int, bool) Method

Sets the value for the specified column using a System.Boolean.

Syntax

Visual Basic

```
Public Sub SetBoolean (
    ByVal colID As Integer,
    ByVal val As Boolean
)
```

C#

```
public void SetBoolean (
    int colID,
    bool val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.5 SetByte(int, byte) Method

Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).

Syntax

Visual Basic

```
Public Sub SetByte (
    ByVal colID As Integer,
    ByVal val As Byte
)
```

C#

```
public void SetByte (
    int colID,
    byte val
```

```
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.6 SetBytes(int, byte[]) Method

Sets the value for the specified column using an array of System.Bytes.

Syntax

Visual Basic

```
Public Sub SetBytes (
    ByVal colID As Integer,
    ByVal val As Byte()
)
```

C#

```
public unsafe void SetBytes (
    int colID,
    byte[] val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

Only suitable for columns of type ULDbType.Binary or ULDbType.LongBinary, or for columns of type ULDbType.UniqueIdentifier when the value is of length 16. The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.7 SetDateTime(int, DateTime) Method

Sets the value for the specified column using a System.DateTime.

Syntax

Visual Basic

```
Public Sub SetDateTime (
    ByVal colID As Integer,
    ByVal val As Date
)
```

C#

```
public unsafe void SetDateTime (
    int colID,
    DateTime val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The set value is accurate to the millisecond. The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]

Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.8 SetDBNull(int) Method

Sets a column to NULL.

Syntax

Visual Basic

```
Public Sub SetDBNull (ByVal colID As Integer)
```

C#

```
public void SetDBNull (int colID)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data is not actually changed until you execute an ULTable.Insert or Update, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
IsColumnNullable(int) Method [page 553]
Insert() Method [page 527]
Update() Method [page 451]

FieldCount Property [page 305]

1.1.26.9 SetDecimal(int, decimal) Method

Sets the value for the specified column using a System.Decimal.

Syntax

Visual Basic

```
Public Sub SetDecimal (
    ByVal colID As Integer,
    ByVal val As Decimal
)
```

C#

```
public void SetDecimal (
    int colID,
    decimal val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]

Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.10 SetDouble(int, double) Method

Sets the value for the specified column using a System.Double.

Syntax

Visual Basic

```
Public Sub SetDouble (
    ByVal colID As Integer,
    ByVal val As Double
)
```

C#

```
public void SetDouble (
    int colID,
    double val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.11 SetFloat(int, float) Method

Sets the value for the specified column using a System.Single.

Syntax

Visual Basic

```
Public Sub SetFloat (
    ByVal colID As Integer,
    ByVal val As Single
)
```

C#

```
public void SetFloat (
    int colID,
    float val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.12 SetGuid(int, Guid) Method

Sets the value for the specified column using a System.Guid.

Syntax

Visual Basic

```
Public Sub SetGuid (
    ByVal colID As Integer,
    ByVal val As Guid
)
```

C#

```
public unsafe void SetGuid (
    int colID,
    Guid val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed. Only valid for columns of type ULDbType.UniqueIdentifier or for columns of type ULDbType.Binary with a length of 16.

Related Information

GetNewUUID() Method [page 141]
GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
GetColumnSize(int) Method [page 219]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.13 SetInt16(int, short) Method

Sets the value for the specified column using a System.Int16.

Syntax

Visual Basic

```
Public Sub SetInt16 (
    ByVal colID As Integer,
    ByVal val As Short
)
```

C#

```
public void SetInt16 (
    int colID,
    short val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.14 SetInt32(int, int) Method

Sets the value for the specified column using a System.Int32.

Syntax

Visual Basic

```
Public Sub SetInt32 (
    ByVal colID As Integer,
    ByVal val As Integer
)
```

C#

```
public void SetInt32 (
    int colID,
    int val
```

```
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.15 SetInt64(int, long) Method

Sets the value for the specified column using an Int64.

Syntax

Visual Basic

```
Public Sub SetInt64 (
    ByVal colID As Integer,
    ByVal val As Long
)
```

C#

```
public void SetInt64 (
    int colID,
    long val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.16 SetString(int, string) Method

Sets the value for the specified column using a System.String.

Syntax

Visual Basic

```
Public Sub SetString (
    ByVal colID As Integer,
    ByVal val As String
)
```

C#

```
public unsafe void SetString (
    int colID,
    string val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]

Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.17 SetTimeSpan(int, TimeSpan) Method

Sets the value for the specified column using a System.TimeSpan.

Syntax

Visual Basic

```
Public Sub SetTimeSpan (
    ByVal colID As Integer,
    ByVal val As TimeSpan
)
```

C#

```
public unsafe void SetTimeSpan (
    int colID,
    TimeSpan val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

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 a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.18 SetToDefault(int) Method

Sets the value for the specified column to its default value.

Syntax

Visual Basic

```
Public Sub SetToDefault (ByVal colID As Integer)
```

C#

```
public void SetToDefault (int colID)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
GetColumnDefaultValue(int) Method [page 541]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.19 SetUInt16(int, ushort) Method

Sets the value for the specified column using a System.UInt16.

Syntax

Visual Basic

```
Public Sub SetUInt16 (
    ByVal colID As Integer,
    ByVal val As UShort
)
```

C#

```
public void SetUInt16 (
    int colID,
    ushort val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.20 SetUInt32(int, uint) Method

Sets the value for the specified column using a System.UInt32.

Syntax

Visual Basic

```
Public Sub SetUInt32 (
    ByVal colID As Integer,
    ByVal val As UInteger
)
```

C#

```
public void SetUInt32 (
    int colID,
    uint val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.21 SetUInt64(int, ulong) Method

Sets the value for the specified column using a System.UInt64.

Syntax

Visual Basic

```
Public Sub SetUInt64 (
    ByVal colID As Integer,
    ByVal val As ULong
)
```

C#

```
public void SetUInt64 (
    int colID,
    ulong val
)
```

Parameters

colID 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.

Exceptions

ULException class A SQL error occurred.

Remarks

The data in the row is not actually changed until you execute a ULTable.Insert or Update method, and changes are not made permanent until committed.

Related Information

GetOrdinal(string) Method [page 286]
Schema Property [page 309]
GetFieldType(int) Method [page 279]
Insert() Method [page 527]
Update() Method [page 451]
FieldCount Property [page 305]

1.1.26.22 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.

Related Information

UpdateBegin() Method [page 452]

1.1.26.23 UpdateBegin() Method

Prepares to update the current row.

Syntax

Visual Basic

```
Public Sub UpdateBegin ()
```

C#

```
public void UpdateBegin ()
```

Exceptions

ULException class A SQL error occurred.

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 the Update method, and changes are not made permanent until committed.

Modifying columns in the index used to open the table affects active searches in unpredictable ways. Columns in the primary key of the table can not be updated.

Related Information

Update() Method [page 451]

1.1.27 ULResultSetSchema Class

UL Ext: Represents the schema of an UltraLite result set.

Syntax

Visual Basic

```
Public NotInheritable Class ULResultSetSchema Inherits ULCursorSchema
```

C#

```
public sealed class ULResultSetSchema : ULCursorSchema
```

Members

All members of ULResultSetSchema, including inherited members.

Methods

Modifier and Type	Method	Description
protected virtual override void	VerifyOpen() [page 455]	

Properties

Modifier and Type	Property	Description
public override string	Name [page 455]	Returns the name of the cursor.

Inherited members from ULCursorSchema

Modifier and Type	Member	Description
public short	ColumnCount [page 223]	Returns the number of columns in the cursor.
protected unsafe void	GetColumnCount() [page 215]	
public unsafe short	GetColumnID(string) [page 215]	Returns the column ID of the named column.
public string	GetColumnName(int) [page 216]	Returns the name of the column identified by the specified column ID.
public unsafe int	GetColumnPrecision(int) [page 217]	Returns the precision of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).
public unsafe int	GetColumnScale(int) [page 218]	Returns the scale of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).

Modifier and Type	Member	Description
public unsafe int	GetColumnSize(int) [page 219]	Returns the size of the column identified by the specified column ID if the column is a sized column (the BINARY or CHAR SQL types).
public string	GetColumnSQLName(int) [page 220]	Returns the name of the column identified by the specified column ID.
public unsafe ULDbType	GetColumnULDbType(int) [page 221]	Returns the UltraLite.NET data type of the column identified by the specified column ID.
public unsafe DataTable	GetSchemaTable() [page 222]	Returns a System.Data.DataTable that describes the column schema of the ULDataReader object.
public bool	IsOpen [page 223]	Checks whether the cursor schema is currently open.

Remarks

There is no constructor for this class. A ULResultSetSchema object is attached to a result set as its ULDataReader.Schema property.

A result set schema is only valid while the data reader is open.

In this section:

VerifyOpen() Method [page 455]

Name Property [page 455]
Returns the name of the cursor.

Related Information

ULCommand Class [page 45]
ULDataReader Class [page 260]
Schema Property [page 309]
ULCursorSchema Class [page 212]

1.1.27.1 VerifyOpen() Method

Syntax

Visual Basic

```
Protected Overridable Overrides Sub VerifyOpen ()
```

C#

```
protected virtual override void VerifyOpen ()
```

1.1.27.2 Name Property

Returns the name of the cursor.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Name As String
```

C#

```
public override string Name {get;}
```

Remarks

The SQL statement that generated the ULResultSetSchema.

1.1.28 ULRowsCopiedEventArgs Class

Represents the set of arguments passed to the ULRowsCopiedEventHandler object.

Syntax

Visual Basic

```
Public NotInheritable Class ULRowsCopiedEventArgs
```

C#

```
public sealed class ULRowsCopiedEventArgs
```

Members

All members of ULRowsCopiedEventArgs, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULRowsCopiedEventArgs(long) [page 457]	Creates a new instance of the ULRowsCopiedEventArgs object.

Properties

Modifier and Type	Property	Description
public bool	Abort [page 457]	Gets or sets a value that indicates whether the bulk-copy operation should be aborted.
public long	RowsCopied [page 458]	Returns the number of rows copied during the current bulk-copy operation.

Remarks

The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

In this section:

ULRowsCopiedEventArgs(long) Constructor [page 457]
Creates a new instance of the ULRowsCopiedEventArgs object.

Abort Property [page 457]
Gets or sets a value that indicates whether the bulk-copy operation should be aborted.

RowsCopied Property [page 458]
Returns the number of rows copied during the current bulk-copy operation.

Related Information

ULRowsCopiedEventHandler(object, ULRowsCopiedEventArgs) Delegate [page 563]

1.1.28.1 ULRowsCopiedEventArgs(long) Constructor

Creates a new instance of the ULRowsCopiedEventArgs object.

Syntax

Visual Basic

```
Public Sub ULRowsCopiedEventArgs (ByVal rowsCopied As Long)
```

C#

```
public ULRowsCopiedEventArgs (long rowsCopied)
```

Parameters

rowsCopied A 64-bit integer value that indicates the number of rows copied during the current bulk-copy operation.

Remarks

The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

1.1.28.2 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

The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

1.1.28.3 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;}
```

Remarks

A long integer representing the number of rows copied.

The ULRowsCopiedEventArgs class is not available in the .NET Compact Framework 2.0.

1.1.29 ULRowUpdatedEventArgs Class

Provides data for the ULDataAdapter.RowUpdated event.

Syntax

Visual Basic

```
Public NotInheritable Class ULRowUpdatedEventArgs Inherits
System.Data.Common.RowUpdatedEventArgs
```

C#

```
public sealed class ULRowUpdatedEventArgs :
System.Data.Common.RowUpdatedEventArgs
```

Members

All members of ULRowUpdatedEventArgs, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULRowUpdatedEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) [page 459]	Initializes a new instance of the ULRowUpdatedEventArgs class.

Properties

Modifier and Type	Property	Description
public new ULCommand	Command [page 460]	Returns the ULCommand object executed when the DbDataAdapter.Update method is called.
public new int	RecordsAffected [page 461]	Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement.

In this section:

ULRowUpdatedEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) Constructor [page 459]
Initializes a new instance of the ULRowUpdatedEventArgs class.

Command Property [page 460]
Returns the ULCommand object executed when the DbDataAdapter.Update method is called.

RecordsAffected Property [page 461]
Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement.

Related Information

RowUpdated Event [page 238]

1.1.29.1 ULRowUpdatedEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) Constructor

Initializes a new instance of the ULRowUpdatedEventArgs class.

Syntax

Visual Basic

```
Public Sub ULRowUpdatedEventArgs (
    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 a DbDataAdapter.Update call.
command The System.Data.IDbCommand executed when the DbDataAdapter.Update method 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 a DbDataAdapter.Update call.

1.1.29.2 Command Property

Returns the ULCommand object executed when the DbDataAdapter.Update method is called.

Syntax

Visual Basic

```
Public ReadOnly Shadows Property Command As ULCommand
```

C#

```
public new ULCommand Command {get;}
```

Remarks

The ULCommand object executed by the update.

This is the strongly-typed version of System.Data.Common.RowUpdatedEventArgs.Command.

Related Information

ULCommand Class [page 45]

1.1.29.3 RecordsAffected Property

Returns the number of rows changed, inserted, or deleted by the execution of the SQL statement.

Syntax

Visual Basic

```
Public ReadOnly Shadows Property RecordsAffected As Integer
```

C#

```
public new int RecordsAffected {get;}
```

Remarks

For SELECT statements this value is -1.

The number of rows changed, inserted, or deleted; 0 if no rows were affected or the statement failed; and -1 for SELECT statements.

1.1.30 ULRowUpdatingEventArgs Class

Provides data for the ULDataAdapter.RowUpdating event.

Syntax

Visual Basic

```
Public NotInheritable Class ULRowUpdatingEventArgs Inherits
System.Data.Common.RowUpdatingEventArgs
```

C#

```
public sealed class ULRowUpdatingEventArgs :
System.Data.Common.RowUpdatingEventArgs
```

Members

All members of ULRowUpdatingEventArgs, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ULRowUpdatingEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) [page 462]	Initializes a new instance of the ULRowUpdatingEventArgs class.

Properties

Modifier and Type	Property	Description
public new ULCommand	Command [page 463]	Specifies the ULCommand object to execute when performing the DbDataAdapter.Update method.

In this section:

ULRowUpdatingEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) Constructor [page 462]
Initializes a new instance of the ULRowUpdatingEventArgs class.

Command Property [page 463]
Specifies the ULCommand object to execute when performing the DbDataAdapter.Update method.

Related Information

RowUpdating Event [page 239]

1.1.30.1 ULRowUpdatingEventArgs(DataRow, IDbCommand, StatementType, DataTableMapping) Constructor

Initializes a new instance of the ULRowUpdatingEventArgs class.

Syntax

Visual Basic

```
Public Sub ULRowUpdatingEventArgs (
    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 value sent through a DbDataAdapter.Update call.

1.1.30.2 Command Property

Specifies the ULCommand object to execute when performing the DbDataAdapter.Update method.

Syntax

Visual Basic

```
Public Shadows Property Command As ULCommand
```

C#

```
public new ULCommand Command {get;set;}
```

Remarks

The ULCommand object to execute when updating.

This is the strongly-typed version of the System.Data.Common.RowUpdatingEventArgs.Command value.

Related Information

ULCommand Class [page 45]

1.1.31 ULServerSyncListener Interface

UL Ext: The listener interface for receiving server synchronization messages.

Syntax

Visual Basic

```
Public Interface ULServerSyncListener
```

C#

```
public interface ULServerSyncListener
```

Members

All members of ULServerSyncListener, including inherited members.

Methods

Modifier and Type	Method	Description
public void	ServerSyncInvoked(string) [page 464]	Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

In this section:

ServerSyncInvoked(string) Method [page 464]
Invoked when the MobiLink Listener for server-initiated synchronizations calls the application to perform synchronization.

1.1.31.1 ServerSyncInvoked(string) 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, use a separate connection and use the lock keyword to access any objects shared with the rest of the application.

Example

Imports Sap.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

The following C# code demonstrates how to receive a server synchronization request and perform a synchronization in the UI thread.

```
using Sap.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();
    }
}
```

1.1.32 ULSqlProgressData Class (Deprecated)

UL Ext: Returns SQL passthrough script progress monitoring data.

Syntax

Microsoft Visual Basic

```
Public Class ULSqlProgressData
```

C#

```
public class ULSqlProgressData
```

Members

All members of ULSqlProgressData, including inherited members.

Properties

Modifier and Type	Property	Description
public long	CurrentScript [page 467]	The index of the scripts executed so far.

Modifier and Type	Property	Description
public long	ScriptCount [page 467]	Returns the number of scripts being executed.
public ULSqlProgressState	State [page 468]	Returns the current progress state.

In this section:

CurrentScript Property [page 467]
The index of the scripts executed so far.

ScriptCount Property [page 467]
Returns the number of scripts being executed.

State Property [page 468]
Returns the current progress state.

1.1.32.1 CurrentScript Property

The index of the scripts executed so far.

Syntax

Visual Basic

```
Public ReadOnly Property CurrentScript As Long
```

C#

```
public long CurrentScript {get;}
```

Remarks

The current index of the scripts being executed.

1.1.32.2 ScriptCount Property

Returns the number of scripts being executed.

Syntax

Visual Basic

```
Public ReadOnly Property ScriptCount As Long
```

C#

```
public long ScriptCount {get;}
```

Remarks

The number of scripts being executed.

1.1.32.3 State Property

Returns the current progress state.

Syntax

Visual Basic

```
Public ReadOnly Property State As ULSqlProgressState
```

C#

```
public ULSqlProgressState State {get;}
```

Remarks

One of the ULSqlProgressState values specifying the current SQL passthrough callback state.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.33 ULSyncParms Class

UL Ext: Represents synchronization parameters that define how to synchronize an UltraLite database.

Syntax

Visual Basic

```
Public NotInheritable Class ULSyncParms
```

C#

```
public sealed class ULSyncParms
```

Members

All members of ULSyncParms, including inherited members.

Methods

Modifier and Type	Method	Description
public void	CopyFrom(ULSyncParms) [page 472]	Copies the properties of the specified ULSyncParms object to this ULSync-Parms object.
public override string	ToString() [page 472]	Returns the string representation of this instance.

Properties

Modifier and Type	Property	Description
public string	AdditionalParms [page 473]	Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.
public string[]	AuthenticationParms [page 473]	Specifies parameters for a custom user authentication script (MobiLink authen-ticate_parameters connection event).
public bool	DownloadOnly [page 474]	Specifies whether to disable or enable uploads when synchronizing.
public bool	KeepPartialDownload [page 475]	Specifies whether to disable or enable partial downloads when synchronizing.
public string	NewPassword [page 476]	Specifies a new MobiLink password for the user specified with UserName.
public string	Password [page 477]	The MobiLink password for the user specified by UserName.
public bool	PingOnly [page 478]	Specifies whether the client should only ping the MobiLink server instead of per-forming a real synchronization.
public string	Publications [page 479]	Specifies the publications to be synchronized.
public bool	ResumePartialDownload [page 479]	Specifies whether to resume or discard a previous partial download.
public bool	SendDownloadAck [page 480]	Specifies whether the client should send a download acknowledgement to the MobiLink server during synchroni-zation.

Modifier and Type	Property	Description
public ULStreamType	Stream [page 481]	Specifies the MobiLink synchronization stream to use for synchronization.
public string	StreamParms [page 481]	Specifies the parameters to configure the synchronization stream.
public bool	UploadOnly [page 482]	Specifies whether to disable or enable downloads when synchronizing.
public string	UserName [page 483]	The user name that uniquely identifies the MobiLink client to the MobiLink server.
public string	Version [page 484]	Specifies which synchronization script to use.

Remarks

There is no constructor for this class. Each connection has its own ULSyncParms instance, attached as its ULConnection.SyncParms property.

At most, only one synchronization command (the ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly property) 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 the ULConnection.Synchronize method.

Other sources of ULSQLCode.SQLE_SYNC_INFO_INVALID errors include not specifying a ULSyncParms.Stream value or a ULSyncParms.Version value.

In this section:

CopyFrom(ULSyncParms) Method [page 472]
Copies the properties of the specified ULSyncParms object to this ULSyncParms object.

ToString() Method [page 472]
Returns the string representation of this instance.

AdditionalParms Property [page 473]
Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.

AuthenticationParms Property [page 473]
Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

DownloadOnly Property [page 474]
Specifies whether to disable or enable uploads when synchronizing.

KeepPartialDownload Property [page 475]
Specifies whether to disable or enable partial downloads when synchronizing.

NewPassword Property [page 476]
Specifies a new MobiLink password for the user specified with UserName.

Password Property [page 477]

The MobiLink password for the user specified by UserName.

PingOnly Property [page 478]
Specifies whether the client should only ping the MobiLink server instead of performing a real synchronization.

Publications Property [page 479]
Specifies the publications to be synchronized.

ResumePartialDownload Property [page 479]
Specifies whether to resume or discard a previous partial download.

SendDownloadAck Property [page 480]
Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization.

Stream Property [page 481]
Specifies the MobiLink synchronization stream to use for synchronization.

StreamParms Property [page 481]
Specifies the parameters to configure the synchronization stream.

UploadOnly Property [page 482]
Specifies whether to disable or enable downloads when synchronizing.

UserName Property [page 483]
The user name that uniquely identifies the MobiLink client to the MobiLink server.

Version Property [page 484]
Specifies which synchronization script to use.

Related Information

ULConnection Class [page 108]
SyncParms Property [page 169]
Synchronize() Method [page 157]
DownloadOnly Property [page 474]
PingOnly Property [page 478]
ResumePartialDownload Property [page 479]
UploadOnly Property [page 482]
Synchronize() Method [page 157]
Stream Property [page 481]
Version Property [page 484]

1.1.33.1 CopyFrom(ULSyncParms) 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.

Related Information

ULSyncParms Class [page 468]

1.1.33.2 ToString() Method

Returns the string representation of this instance.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

Returns

The string representation of this instance as a semicolon-separated list of keyword=value pairs.

1.1.33.3 AdditionalParms Property

Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.

Syntax

Visual Basic

```
Public Property AdditionalParms As String
```

C#

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

Returns

A string, in the form of a semicolon-separated list of name=value pairs.

Remarks

Use this property to specify several additional synchronization parameters that cannot be readily specified using any other predefined parameters.

Example

```
private ULSyncParms info;
// ...
info.AdditionalParms =
   "AllowDownloadDupRows=1;
   CheckpointStore=1;
   DisableConcurrency=1;
   TableOrder=Customer,Sales"
```

1.1.33.4 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;}
```

Returns

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 the MobiLink server's limit for authentication parameters (currently 4000 UTF8 bytes).

1.1.33.5 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;}
```

Returns

True to disable uploads when synchronizing, false to enable uploads. The default is false.

Remarks

At most, only one synchronization command (the ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly property) 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 the ULConnection.Synchronize method.

Related Information

UploadOnly Property [page 482]
PingOnly Property [page 478]
ResumePartialDownload Property [page 479]
UploadOnly Property [page 482]
Synchronize() Method [page 157]

1.1.33.6 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;}
```

Returns

Set to true to enable and save partial downloads while synchronizing; otherwise, set to false to disable partial downloads and roll back downloads if any errors occur. The default is false.

Remarks

Using the ULSyncProgressListener object, UltraLite.NET can resume partial downloads that fail because of communication errors or user aborts. 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.

If a partial download was kept, then the ULConnection.ULSyncResult.PartialDownloadRetained property is set to true when the ULConnection.Synchronize method exits.

If the PartialDownloadRetained property is set, then you can resume a download. To do this, call the ULConnection.Synchronize method with the ULConnection.ULSyncParms.ResumePartialDownload property set to true. Keep the KeepPartialDownload property set to true 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 recent data, then you can do another download immediately after the resumed download completes.

When resuming a download, many of the ULSyncParms properties are not relevant. For example, the Publications property is not used. You receive the publications that you requested on the initial download. The only properties that must be set are ResumePartialDownload and UserName. The KeepPartialDownload property can be set if desired and functions as normal.

If you have a partial download that is no longer needed, call the ULConnection.RollbackPartialDownload method to roll back the failed download transaction. If you attempt to synchronize again and do not specify the ResumePartialDownload property, then the partial download is rolled back before the next synchronization begins.

Related Information

PartialDownloadRetained Property [page 504]
ResumePartialDownload Property [page 479]
RollbackPartialDownload() Method [page 152]
ResumePartialDownload Property [page 479]
UserName Property [page 483]
RollbackPartialDownload() Method [page 152]

1.1.33.7 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;}
```

Returns

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.

Related Information

UserName Property [page 483]

1.1.33.8 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;}
```

Returns

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.

Related Information

NewPassword Property [page 476]
UserName Property [page 483]

1.1.33.9 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;}
```

Returns

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 (the ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly property) 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 the ULConnection.Synchronize method.

Related Information

DownloadOnly Property [page 474]
ResumePartialDownload Property [page 479]
UploadOnly Property [page 482]
Synchronize() Method [page 157]

1.1.33.10 Publications Property

Specifies the publications to be synchronized.

Syntax

Visual Basic

```
Public Property Publications As String
```

C#

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

Returns

A string containing a list of publication names, separated by comma (,); or the special value ULConnection.SYNC_ALL_PUBS, or the special value ULConnection.SYNC_ALL_DB. The default is ULConnection.SYNC_ALL_DB.

1.1.33.11 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;}
```

Returns

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 (the ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly property) 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 the ULConnection.Synchronize method.

Related Information

KeepPartialDownload Property [page 475]
DownloadOnly Property [page 474]
PingOnly Property [page 478]
UploadOnly Property [page 482]
Synchronize() Method [page 157]
PartialDownloadRetained Property [page 504]

1.1.33.12 SendDownloadAck Property

Specifies whether the client should send a download acknowledgement to the MobiLink server during synchronization.

Syntax

Visual Basic

```
Public Property SendDownloadAck As Boolean
```

C#

```
public bool SendDownloadAck {get;set;}
```

Returns

Set to true to specify that the client should send a download acknowledgement to the MobiLink server. Set to false to specify that no download acknowledgement is sent. The default is false.

Remarks

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).

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.

1.1.33.13 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;}
```

Returns

One of the ULStreamType values specifying the type of synchronization stream to use. The default value 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 ULSyncParms.StreamParms property.

If the stream type is set to a value that is invalid for the platform, the stream type is set to ULStreamType.TCPIP.

Related Information

ULStreamType Enumeration [page 574]
StreamParms Property [page 481]

1.1.33.14 StreamParms Property

Specifies the parameters to configure the synchronization stream.

Syntax

Visual Basic

```
Public Property StreamParms As String
```

C#

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

Returns

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

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").

Related Information

Stream Property [page 481]
ULStreamType Enumeration [page 574]

1.1.33.15 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;}
```

Returns

True to disable downloads, false to enable downloads. The default is false.

Remarks

At most, only one synchronization command (the ULSyncParms.DownloadOnly, ULSyncParms.PingOnly, ULSyncParms.ResumePartialDownload, or ULSyncParms.UploadOnly property) 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 the ULConnection.Synchronize method.

Related Information

DownloadOnly Property [page 474]
PingOnly Property [page 478]
ResumePartialDownload Property [page 479]
Synchronize() Method [page 157]

1.1.33.16 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;}
```

Returns

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.

Related Information

Password Property [page 477]

1.1.33.17 Version Property

Specifies which synchronization script to use.

Syntax

Visual Basic

```
Public Property Version As String
```

C#

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

Returns

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.

1.1.34 ULSyncProgressData Class

UL Ext: Returns synchronization progress monitoring data.

Syntax

Visual Basic

```
Public Class ULSyncProgressData
```

C#

```
public class ULSyncProgressData
```

Members

All members of ULSyncProgressData, including inherited members.

Variables

Modifier and Type	Variable	Description
public const int	FLAG_IS_BLOCKING	A flag indicating that the synchronization is blocked awaiting a response from the MobiLink server.
public const int	FLAG_LAST_UPLOAD_RECEIVED	A flag indicating whether the server received the last upload. If it did not, the upload will be resent the next time the publications from the previous sync are synced again, which could be during the current synchronization.

Properties

Modifier and Type	Property	Description
public int	CurrentDownloadRowCount [page 487]	Returns the number of rows that have been downloaded so far.
public int	Flags [page 488]	Returns the current synchronization flags indicating additional information relating to the current state.
public int	IgnoredDeletes [page 489]	Returns the number of rows received so far that have already been deleted.
public int	IgnoredUpdates [page 489]	Returns the number of rows received so far that have already been updated.
public bool	IsFinalSyncProgress [page 490]	Returns true if this is final synchronization progress message.
public long	ReceivedBytes [page 490]	Returns the number of bytes received so far.
public int	ReceivedDeletes [page 491]	Returns the number of deleted rows received so far.
public int	ReceivedInserts [page 491]	Returns the number of inserted rows received so far.
public int	ReceivedUpdates [page 492]	Returns the number of updated rows received so far.
public long	SentBytes [page 493]	Returns the number of bytes sent so far.
public int	SentDeletes [page 493]	Returns the number of deleted rows sent so far.
public int	SentInserts [page 494]	Returns the number of inserted rows sent so far.

Modifier and Type	Property	Description
public int	SentUpdates [page 494]	Returns the number of updated rows sent so far.
public ULSyncProgressState	State [page 495]	Returns the current synchronization state.
public int	SyncTableCount [page 496]	Returns the number of tables being synchronized.
public int	SyncTableIndex [page 496]	Returns the index of the table currently being synchronized in the range from 1 to the total number of tables involved with the synchronization.
public int	TableID [page 497]	Returns the database index of the table currently being synchronized.
public string	TableName [page 497]	Returns the name of the current table being uploaded or downloaded.
public int	TotalDownloadRowCount [page 498]	Returns the total number of rows to be received in the download.
public int	TruncateDeletes [page 499]	Returns the number of rows that have been deleted by a truncate operation.

In this section:

CurrentDownloadRowCount Property [page 487]
Returns the number of rows that have been downloaded so far.

Flags Property [page 488]
Returns the current synchronization flags indicating additional information relating to the current state.

IgnoredDeletes Property [page 489]
Returns the number of rows received so far that have already been deleted.

IgnoredUpdates Property [page 489]
Returns the number of rows received so far that have already been updated.

IsFinalSyncProgress Property [page 490]
Returns true if this is final synchronization progress message.

ReceivedBytes Property [page 490]
Returns the number of bytes received so far.

ReceivedDeletes Property [page 491]
Returns the number of deleted rows received so far.

ReceivedInserts Property [page 491]
Returns the number of inserted rows received so far.

ReceivedUpdates Property [page 492]
Returns the number of updated rows received so far.

SentBytes Property [page 493]
Returns the number of bytes sent so far.

SentDeletes Property [page 493]

Returns the number of deleted rows sent so far.

SentInserts Property [page 494]
Returns the number of inserted rows sent so far.

SentUpdates Property [page 494]
Returns the number of updated rows sent so far.

State Property [page 495]
Returns the current synchronization state.

SyncTableCount Property [page 496]
Returns the number of tables being synchronized.

SyncTableIndex Property [page 496]
Returns the index of the table currently being synchronized in the range from 1 to the total number of tables involved with the synchronization.

TableID Property [page 497]
Returns the database index of the table currently being synchronized.

TableName Property [page 497]
Returns the name of the current table being uploaded or downloaded.

TotalDownloadRowCount Property [page 498]
Returns the total number of rows to be received in the download.

TruncateDeletes Property [page 499]
Returns the number of rows that have been deleted by a truncate operation.

Related Information

ULSyncProgressListener Interface [page 499]

1.1.34.1 CurrentDownloadRowCount Property

Returns the number of rows that have been downloaded so far.

Syntax

Visual Basic

```
Public ReadOnly Property CurrentDownloadRowCount As Integer
```

C#

```
public int CurrentDownloadRowCount {get;}
```

Returns

The number of rows that have been downloaded so far.

Remarks

This number includes duplicate rows that aren't included in ReceivedInserts, ReceivedUpdates, or ReceivedDeletes.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.2 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;}
```

Returns

An integer containing a combination of flags or'ed together.

1.1.34.3 IgnoredDeletes Property

Returns the number of rows received so far that have already been deleted.

Syntax

Visual Basic

```
Public ReadOnly Property IgnoredDeletes As Integer
```

C#

```
public int IgnoredDeletes {get;}
```

Returns

The number of rows received so far that have already been deleted.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.4 IgnoredUpdates Property

Returns the number of rows received so far that have already been updated.

Syntax

Visual Basic

```
Public ReadOnly Property IgnoredUpdates As Integer
```

C#

```
public int IgnoredUpdates {get;}
```

Returns

The number of rows received so far that have already been updated.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.5 IsFinalSyncProgress Property

Returns true if this is final synchronization progress message.

Syntax

Visual Basic

```
Public ReadOnly Property IsFinalSyncProgress As Boolean
```

C#

```
public bool IsFinalSyncProgress {get;}
```

Returns

True if this is the final synchronization progress message.

1.1.34.6 ReceivedBytes Property

Returns the number of bytes received so far.

Syntax

Visual Basic

```
Public ReadOnly Property ReceivedBytes As Long
```

C#

```
public long ReceivedBytes {get;}
```

Returns

The number of bytes received so far.

Remarks

This information is updated for all states.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.7 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;}
```

Returns

The number of deleted rows received so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.8 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;}
```

Returns

The number of inserted rows received so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.9 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;}
```

Returns

The number of updated rows received so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.10 SentBytes Property

Returns the number of bytes sent so far.

Syntax

Visual Basic

```
Public ReadOnly Property SentBytes As Long
```

C#

```
public long SentBytes {get;}
```

Returns

The number of bytes sent so far.

Remarks

This information is updated for all states.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.11 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;}
```

Returns

The number of deleted rows sent so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.12 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;}
```

Returns

The number of inserted rows sent so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.13 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;}
```

Returns

The number of updated rows sent so far.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.14 State Property

Returns the current synchronization state.

Syntax

Visual Basic

```
Public ReadOnly Property State As ULSyncProgressState
```

C#

```
public ULSyncProgressState State {get;}
```

Returns

One of the ULSyncProgressState values specifying the current synchronization state.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.15 SyncTableCount Property

Returns the number of tables being synchronized.

Syntax

Visual Basic

```
Public ReadOnly Property SyncTableCount As Integer
```

C#

```
public int SyncTableCount {get;}
```

Returns

The number of tables being synchronized. For each table there is a sending and receiving phase, so this number may be more than the number of tables being synchronized.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.16 SyncTableIndex Property

Returns the index of the table currently being synchronized in the range from 1 to the total number of tables involved with the synchronization.

Syntax

Visual Basic

```
Public ReadOnly Property SyncTableIndex As Integer
```

C#

```
public int SyncTableIndex {get;}
```

Returns

The index of the table currently being synchronized in the range from 1 to the SyncTableCount property value.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.17 TableID Property

Returns the database index of the table currently being synchronized.

Syntax

Visual Basic

```
Public ReadOnly Property TableID As Integer
```

C#

```
public int TableID {get;}
```

Returns

The database index, in the range from 1 to the ULDatabaseSchema.TableCount property value.

Related Information

ULSyncProgressState Enumeration [page 575]
TableCount Property [page 259]

1.1.34.18 TableName Property

Returns the name of the current table being uploaded or downloaded.

Syntax

Visual Basic

```
Public ReadOnly Property TableName As String
```

C#

```
public string TableName {get;}
```

Returns

Name of the current table being synchronized; null if not applicable.

1.1.34.19 TotalDownloadRowCount Property

Returns the total number of rows to be received in the download.

Syntax

Visual Basic

```
Public ReadOnly Property TotalDownloadRowCount As Integer
```

C#

```
public int TotalDownloadRowCount {get;}
```

Returns

The number of rows to be received in the download.

Remarks

This number includes duplicate rows that aren't included in ReceivedInserts, ReceivedUpdates, or ReceivedDeletes. This value isn't set until the synchronization enters the STATE_RECEIVING_TABLE state for the first table.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.34.20 TruncateDeletes Property

Returns the number of rows that have been deleted by a truncate operation.

Syntax

Visual Basic

```
Public ReadOnly Property TruncateDeletes As Integer
```

C#

```
public int TruncateDeletes {get;}
```

Returns

The number of rows deleted by a truncate operation.

Related Information

ULSyncProgressState Enumeration [page 575]

1.1.35 ULSyncProgressListener Interface

UL Ext: The listener interface for receiving synchronization progress events.

Syntax

Visual Basic

```
Public Interface ULSyncProgressListener
```

C#

```
public interface ULSyncProgressListener
```

Members

All members of ULSyncProgressListener, including inherited members.

Methods

Modifier and Type	Method	Description
public bool	SyncProgressed(ULSyncProgressData) [page 500]	Invoked during synchronization to inform the user of progress.

In this section:

SyncProgressed(ULSyncProgressData) Method [page 500]
Invoked during synchronization to inform the user of progress.

Related Information

Synchronize(ULSyncProgressListener) Method [page 158]

1.1.35.1 SyncProgressed(ULSyncProgressData) Method

Invoked during synchronization to inform the user of progress.

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.

Returns

This method should return true to cancel synchronization or return false to continue.

Remarks

This method should return true to cancel synchronization or return false to continue.

No UltraLite.NET API methods should be invoked during a SyncProgressed call.

Related Information

ULSyncProgressData Class [page 484]

1.1.36 ULSyncResult Class

UL Ext: Represents the status of the last synchronization.

Syntax

Visual Basic

```
Public Class ULSyncResult
```

C#

```
public class ULSyncResult
```

Members

All members of ULSyncResult, including inherited members.

Properties

Modifier and Type	Property	Description
public ULAuthStatusCode	AuthStatus [page 503]	Returns the authorization status code for the last synchronization attempt.
public long	AuthValue [page 503]	Returns the return value from custom user authentication synchronization scripts.
public bool	IgnoredRows [page 504]	Checks whether any uploaded rows were ignored during the last synchronization.
public bool	PartialDownloadRetained [page 504]	Checks whether a partial download was retained during the last synchronization.

Modifier and Type	Property	Description
public ULStreamErrorCode	StreamErrorCode [page 505]	Returns the error reported by the stream itself.
public string	StreamErrorParameters [page 505]	Returns a comma-separated list of stream error parameters.
public int	StreamErrorSystem [page 506]	Returns the stream error system-specific code.
public DateTime	Timestamp [page 506]	Returns the timestamp of the last synchronization.
public bool	UploadOK [page 507]	Checks whether the last upload synchronization was successful.

Remarks

There is no constructor for this class. Each connection has its own ULSyncResult instance, attached as its ULConnection.SyncResult property. A ULSyncResult instance is only valid while that connection is open.

In this section:

AuthStatus Property [page 503]
Returns the authorization status code for the last synchronization attempt.

AuthValue Property [page 503]
Returns the return value from custom user authentication synchronization scripts.

IgnoredRows Property [page 504]
Checks whether any uploaded rows were ignored during the last synchronization.

PartialDownloadRetained Property [page 504]
Checks whether a partial download was retained during the last synchronization.

StreamErrorCode Property [page 505]
Returns the error reported by the stream itself.

StreamErrorParameters Property [page 505]
Returns a comma-separated list of stream error parameters.

StreamErrorSystem Property [page 506]
Returns the stream error system-specific code.

Timestamp Property [page 506]
Returns the timestamp of the last synchronization.

UploadOK Property [page 507]
Checks whether the last upload synchronization was successful.

Related Information

SyncResult Property [page 170]
Synchronize() Method [page 157]

1.1.36.1 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;}
```

Remarks

One of the ULAuthStatusCode values denoting the authorization status for the last synchronization attempt.

Related Information

ULAuthStatusCode Enumeration [page 566]

1.1.36.2 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;}
```

Remarks

A long integer returned from custom user authentication synchronization scripts.

1.1.36.3 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;}
```

Remarks

True if any uploaded rows were ignored during the last synchronization, false if no rows were ignored.

Related Information

DownloadOnly Property [page 474]

1.1.36.4 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;}
```

Remarks

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.

Related Information

KeepPartialDownload Property [page 475]

1.1.36.5 StreamErrorCode Property

Returns the error reported by the stream itself.

Syntax

Visual Basic

```
Public ReadOnly Property StreamErrorCode As ULStreamErrorCode
```

C#

```
public ULStreamErrorCode StreamErrorCode {get;}
```

Remarks

One of the ULStreamErrorCode values denoting the error reported by the stream itself, ULStreamErrorCode.NONE if no error occurred.

1.1.36.6 StreamErrorParameters Property

Returns a comma-separated list of stream error parameters.

Syntax

Visual Basic

```
Public ReadOnly Property StreamErrorParameters As String
```

C#

```
public string StreamErrorParameters {get;}
```

Remarks

Contains a comma separated list of error parameters for the stream error code reported in StreamErrorCode property. This is an empty string either for errors with no parameters, or when no error has been set.

Related Information

StreamErrorCode Property [page 337]

1.1.36.7 StreamErrorSystem Property

Returns the stream error system-specific code.

Syntax

Visual Basic

```
Public ReadOnly Property StreamErrorSystem As Integer
```

C#

```
public int StreamErrorSystem {get;}
```

Remarks

An integer denoting the stream error system-specific code.

1.1.36.8 Timestamp Property

Returns the timestamp of the last synchronization.

Syntax

Visual Basic

```
Public ReadOnly Property Timestamp As Date
```

C#

```
public DateTime Timestamp {get;}
```

Remarks

A System.DateTime structure specifying the timestamp of the last synchronization.

1.1.36.9 UploadOK Property

Checks whether the last upload synchronization was successful.

Syntax

Visual Basic

```
Public ReadOnly Property UploadOK As Boolean
```

C#

```
public bool UploadOK {get;}
```

Remarks

True if the last upload synchronization was successful, false if the last upload synchronization was unsuccessful.

1.1.37 ULTable Class

UL Ext: Represents a table in an UltraLite database.

Syntax

Visual Basic

```
Public Class ULTable Inherits ULResultSet
```

C#

```
public class ULTable : ULResultSet
```

Members

All members of ULTable, including inherited members.

Methods

Modifier and Type	Method	Description
public void	DeleteAllRows() [page 514]	Deletes all rows in the table.
public void	FindBegin() [page 515]	Prepares to perform a new Find on a table.
public bool	FindFirst [page 516]	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.
public bool	FindLast [page 519]	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.
public bool	FindNext [page 522]	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.
public bool	FindPrevious [page 525]	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.
public void	Insert() [page 527]	Inserts a new row with the current column values (specified using the set methods).
public void	InsertBegin() [page 528]	Prepares to insert a new row into the table by setting all current column values to their default values.
public bool	LookupBackward [page 529]	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.
public void	LookupBegin() [page 532]	Prepares to perform a new lookup on the table.
public bool	LookupForward [page 533]	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.
public void	Truncate() [page 535]	Deletes all rows in the table while temporarily activating a stop synchronization delete.

Properties

Modifier and Type	Property	Description
public new ULTableSchema	Schema [page 536]	Holds the table schema.

Inherited members from ULResultSet

Modifier and Type	Member	Description
public unsafe void	AppendBytes(int, byte[], int, int) [page 428]	Appends the specified subset of the specified array of System.Bytes to the new value for the specified ULDbType.LongBinary column.
public unsafe void	AppendChars(int, char[], int, int) [page 429]	Appends the specified subset of the specified array of System.Chars to the new value for the specified ULDbType.LongVarchar column.
public void	Delete() [page 431]	Deletes the current row.
public void	SetBoolean(int, bool) [page 431]	Sets the value for the specified column using a System.Boolean.
public void	SetByte(int, byte) [page 432]	Sets the value for the specified column using a System.Byte (unsigned 8-bit integer).
public unsafe void	SetBytes(int, byte[]) [page 433]	Sets the value for the specified column using an array of System.Bytes.
public unsafe void	SetDateTime(int, DateTime) [page 435]	Sets the value for the specified column using a System.DateTime.
public void	SetDBNull(int) [page 436]	Sets a column to NULL.
public void	SetDecimal(int, decimal) [page 437]	Sets the value for the specified column using a System.Decimal.
public void	SetDouble(int, double) [page 438]	Sets the value for the specified column using a System.Double.
public void	SetFloat(int, float) [page 439]	Sets the value for the specified column using a System.Single.
public unsafe void	SetGuid(int, Guid) [page 440]	Sets the value for the specified column using a System.Guid.
public void	SetInt16(int, short) [page 441]	Sets the value for the specified column using a System.Int16.
public void	SetInt32(int, int) [page 442]	Sets the value for the specified column using a System.Int32.
public void	SetInt64(int, long) [page 443]	Sets the value for the specified column using an Int64.
public unsafe void	SetString(int, string) [page 445]	Sets the value for the specified column using a System.String.
public unsafe void	SetTimeSpan(int, TimeSpan) [page 446]	Sets the value for the specified column using a System.TimeSpan.
public void	SetToDefault(int) [page 447]	Sets the value for the specified column to its default value.
public void	SetUInt16(int, ushort) [page 448]	Sets the value for the specified column using a System.UInt16.

Modifier and Type	Member	Description
public void	SetUInt32(int, uint) [page 449]	Sets the value for the specified column using a System.UInt32.
public void	SetUInt64(int, ulong) [page 450]	Sets the value for the specified column using a System.UInt64.
public void	Update() [page 451]	Updates the current row with the current column values (specified using the set methods).
public void	UpdateBegin() [page 452]	Prepares to update the current row.

Inherited members from ULDataReader

Modifier and Type	Member	Description
public override void	Close() [page 266]	Closes the cursor.
public override int	Depth [page 305]	Returns the depth of nesting for the current row.
protected override void	Dispose(bool) [page 267]	
public override int	FieldCount [page 305]	Returns the number of columns in the cursor.
public override unsafe bool	GetBoolean(int) [page 267]	Returns the value for the specified column as a System.Boolean.
public override unsafe byte	GetByte(int) [page 268]	Returns the value for the specified column as an unsigned 8-bit value (System.Byte).
public unsafe byte[]	GetBytes(int) [page 269]	UL Ext: Returns the value for the specified column as an array of System.Bytes values.
public override unsafe long	GetBytes(int, long, byte[], int, int) [page 270]	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.
public override char	GetChar(int) [page 272]	This method is not supported in UltraLite.NET.
public override unsafe long	GetChars(int, long, char[], int, int) [page 273]	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.
public override string	GetDataTypeName(int) [page 274]	Returns the name of the specified column's provider data type.
public override unsafe DateTime	GetDateTime(int) [page 275]	Returns the value for the specified column as a System.DateTime type with millisecond accuracy.
protected override DbDataReader	GetDbDataReader(int) [page 276]	

Modifier and Type	Member	Description
public override decimal	GetDecimal(int) [page 276]	Returns the value for the specified column as a System.Decimal type.
public override unsafe double	GetDouble(int) [page 277]	Returns the value for the specified column as a System.Double type.
public override IEnumerator	GetEnumerator() [page 278]	Returns an System.Collections.IEnumerator value that iterates through the ULDataReader object.
public override Type	GetFieldType(int) [page 279]	Returns the System.Type value most appropriate for the specified column.
public override unsafe float	GetFloat(int) [page 280]	Returns the value for the specified column as a System.Single type.
public override unsafe Guid	GetGuid(int) [page 281]	Returns the value for the specified column as a UUID (System.Guid) type.
public override unsafe short	GetInt16(int) [page 282]	Returns the value for the specified column as a System.Int16 type.
public override unsafe int	GetInt32(int) [page 283]	Returns the value for the specified column as a System.Int32 type.
public override unsafe long	GetInt64(int) [page 284]	Returns the value for the specified column as a System.Int64 type.
public override string	GetName(int) [page 285]	Returns the name of the specified column.
public override unsafe int	GetOrdinal(string) [page 286]	Returns the column ID of the named column.
public unsafe int	GetRowCount(int) [page 287]	UL Ext: Returns the number of rows in the cursor, within threshold.
public override DataTable	GetSchemaTable() [page 288]	Returns a System.Data.DataTable value that describes the column metadata of the ULDataReader object.
public override unsafe String	GetString(int) [page 290]	Returns the value for the specified column as a System.String type.
public unsafe TimeSpan	GetTimeSpan(int) [page 291]	Returns the value for the specified column as a System.TimeSpan type with millisecond accuracy.
public unsafe ushort	GetUInt16(int) [page 292]	Returns the value for the specified column as a System.UInt16 type.
public unsafe uint	GetUInt32(int) [page 293]	Returns the value for the specified column as a System.UInt32 type.
public unsafe ulong	GetUInt64(int) [page 294]	Returns the value for the specified column as a System.UInt64 type.
public override object	GetValue(int) [page 295]	Returns the value of the specified column in its native format.
public override int	GetValues(object[]) [page 296]	Returns all the column values for the current row.

Modifier and Type	Member	Description
public override unsafe bool	HasRows [page 306]	Checks whether the ULDataReader object has one or more rows.
public unsafe bool	IsBOF [page 306]	UL Ext: Checks whether the current row position is before the first row.
public override bool	IsClosed [page 307]	Checks whether the cursor is currently open.
public override unsafe bool	IsDBNull(int) [page 297]	Checks whether the value from the specified column is NULL.
public unsafe bool	IsEOF [page 307]	UL Ext: Checks whether the current row position is after the last row.
public void	MoveAfterLast() [page 298]	UL Ext: Positions the cursor to after the last row of the cursor.
public void	MoveBeforeFirst() [page 298]	UL Ext: Positions the cursor to before the first row of the cursor.
public unsafe bool	MoveFirst() [page 299]	UL Ext: Positions the cursor to the first row of the cursor.
public unsafe bool	MoveLast() [page 299]	UL Ext: Positions the cursor to the last row of the cursor.
public unsafe bool	MoveNext() [page 300]	UL Ext: Positions the cursor to the next row or after the last row if the cursor was already on the last row.
public unsafe bool	MovePrevious() [page 301]	UL Ext: Positions the cursor to the previous row or before the first row.
public unsafe bool	MoveRelative(int) [page 301]	UL Ext: Positions the cursor relative to the current row.
public override bool	NextResult() [page 302]	Advances the ULDataReader object to the next result when reading the results of batch SQL statements.
public override bool	Read() [page 303]	Positions the cursor to the next row, or after the last row if the cursor was already on the last row.
public override int	RecordsAffected [page 308]	Returns the number of rows changed, inserted, or deleted by execution of the SQL statement.
public int	RowCount [page 308]	UL Ext: Returns the number of rows in the cursor.
public ULCursorSchema	Schema [page 309]	UL Ext: Holds the schema of this cursor.
public override object	this[int colID] [page 310]	Returns the value of the specified column in its native format.
public override object	this[string name] [page 311]	Returns the value of the specified named column in its native format.
protected void	Validate() [page 304]	
protected void	Validate(int) [page 304]	

Remarks

There is no constructor for this class. Tables are created using the ULCommand.ExecuteTable method.

In this section:

DeleteAllRows() Method [page 514]
Deletes all rows in the table.

FindBegin() Method [page 515]
Prepares to perform a new Find on a table.

FindFirst Method [page 516]
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 Method [page 519]
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.

FindNext Method [page 522]
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 Method [page 525]
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.

Insert() Method [page 527]
Inserts a new row with the current column values (specified using the set methods).

InsertBegin() Method [page 528]
Prepares to insert a new row into the table by setting all current column values to their default values.

LookupBackward Method [page 529]
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 [page 532]
Prepares to perform a new lookup on the table.

LookupForward Method [page 533]
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.

Truncate() Method [page 535]
Deletes all rows in the table while temporarily activating a stop synchronization delete.

Schema Property [page 536]
Holds the table schema.

Related Information

ExecuteTable() Method [page 81]

ULCommand Class [page 45]
ULResultSet Class [page 421]

1.1.37.1 DeleteAllRows() Method

Deletes all rows in the table.

Syntax

Visual Basic

```
Public Sub DeleteAllRows ()
```

C#

```
public void DeleteAllRows ()
```

Exceptions

ULException class A SQL error occurred.

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 the ULConnection.StopSynchronizationDelete method.

Related Information

Truncate() Method [page 535]
StopSynchronizationDelete() Method [page 156]

1.1.37.2 FindBegin() Method

Prepares to perform a new Find on a table.

Syntax

Visual Basic

```
Public Sub FindBegin ()
```

C#

```
public void FindBegin ()
```

Exceptions

ULException class A SQL error occurred.

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.

Related Information

FindFirst() Method [page 516]
FindFirst(short) Method [page 517]
FindLast() Method [page 519]
FindLast(short) Method [page 520]

1.1.37.3 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.

Overload List

Modifier and Type	Overload name	Description
public bool	FindFirst() [page 516]	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.
public unsafe bool	FindFirst(short) [page 517]	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.

In this section:

FindFirst() Method [page 516]
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(short) Method [page 517]
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.

1.1.37.3.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

The FindBegin method must be called before each search.

Related Information

FindBegin() Method [page 515]
FindNext() Method [page 522]
FindPrevious() Method [page 525]
FindFirst(short) Method [page 517]
IsEOF Property [page 307]
FindBegin() Method [page 515]

1.1.37.3.2 FindFirst(short) 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 unsafe 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.

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

The FindBegin method must be called before each search.

Related Information

FindBegin() Method [page 515]
FindNext(short) Method [page 523]
FindPrevious(short) Method [page 526]
FindFirst() Method [page 516]
IsEOF Property [page 307]
FindBegin() Method [page 515]

1.1.37.4 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.

Overload List

Modifier and Type	Overload name	Description
public bool	FindLast() [page 519]	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.
public unsafe bool	FindLast(short) [page 520]	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.

In this section:

FindLast() Method [page 519]
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(short) Method [page 520]
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.

1.1.37.4.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

The FindBegin method must be called before each search.

Related Information

FindBegin() Method [page 515]
FindNext() Method [page 522]
FindPrevious() Method [page 525]
FindLast(short) Method [page 520]
IsBOF Property [page 306]
FindBegin() Method [page 515]

1.1.37.4.2 FindLast(short) 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 unsafe 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.

Returns

True if successful, false otherwise

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

The FindBegin method must be called before each search.

Related Information

FindBegin() Method [page 515]
FindNext(short) Method [page 523]
FindPrevious(short) Method [page 526]
FindLast() Method [page 519]
IsBOF Property [page 306]
FindBegin() Method [page 515]

1.1.37.5 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.

Overload List

Modifier and Type	Overload name	Description
public bool	FindNext() [page 522]	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.
public unsafe bool	FindNext(short) [page 523]	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.

In this section:

FindNext() Method [page 522]
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(short) Method [page 523]
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.

1.1.37.5.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

FindNext method behavior is undefined if the column values being searched for are modified during a row update.

Related Information

FindFirst() Method [page 516]
FindNext(short) Method [page 523]
IsEOF Property [page 307]

1.1.37.5.2 FindNext(short) 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 unsafe 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.

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

FindNext method behavior is undefined if the column values being searched for are modified during a row update.

Related Information

FindFirst(short) Method [page 517]
FindNext() Method [page 522]
FindFirst() Method [page 516]
IsEOF Property [page 307]

1.1.37.6 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.

Overload List

Modifier and Type	Overload name	Description
public bool	FindPrevious() [page 525]	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.
public unsafe bool	FindPrevious(short) [page 526]	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.

In this section:

FindPrevious() Method [page 525]
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(short) Method [page 526]
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.

1.1.37.6.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

FindPrevious method behavior is undefined if the column values being searched for are modified during a row update.

Related Information

FindLast() Method [page 519]
FindPrevious(short) Method [page 526]
IsBOF Property [page 306]

1.1.37.6.2 FindPrevious(short) 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 unsafe 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.

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

FindPrevious method behavior is undefined if the column values being searched for are modified during a row update.

Related Information

FindLast() Method [page 519]
FindLast(short) Method [page 520]
FindPrevious() Method [page 525]
IsBOF Property [page 306]

1.1.37.7 Insert() Method

Inserts a new row with the current column values (specified using the set methods).

Syntax

Visual Basic

```
Public Sub Insert ()
```

C#

```
public void Insert ()
```

Exceptions

ULException class A SQL error occurred.

Remarks

Each insert must be preceded by a call to the ULTable.InsertBegin method.

Related Information

InsertBegin() Method [page 528]

1.1.37.8 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 ()
```

Exceptions

ULException class A SQL error occurred.

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.

Related Information

Insert() Method [page 527]
Insert() Method [page 527]

1.1.37.9 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.

Overload List

Modifier and Type	Overload name	Description
public bool	LookupBackward() [page 530]	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.
public unsafe bool	LookupBackward(short) [page 531]	Moves backward through the table from the end, looking for a row that matches or is less than a value or partial set of values in the current index.

In this section:

LookupBackward() Method [page 530]
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(short) Method [page 531]
Moves backward through the table from the end, looking for a row that matches or is less than a value or partial set of values in the current index.

1.1.37.9.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

The LookupBegin method must be called before each search.

Related Information

LookupBegin() Method [page 532]
LookupBackward(short) Method [page 531]
IsBOF Property [page 306]

1.1.37.9.2 LookupBackward(short) Method

Moves backward through the table from the end, 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 unsafe 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.

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsBOF property).

The LookupBegin method must be called before each search.

Related Information

LookupBegin() Method [page 532]
IsBOF Property [page 306]

1.1.37.10 LookupBegin() Method

Prepares to perform a new lookup on the table.

Syntax

Visual Basic

```
Public Sub LookupBegin ()
```

C#

```
public void LookupBegin ()
```

Exceptions

ULException class A SQL error occurred.

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.

Related Information

LookupForward() Method [page 533]
LookupForward(short) Method [page 534]
LookupBackward() Method [page 530]
LookupBackward(short) Method [page 531]

1.1.37.11 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.

Overload List

Modifier and Type	Overload name	Description
public bool	LookupForward() [page 533]	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.
public unsafe bool	LookupForward(short) [page 534]	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.

In this section:

LookupForward() Method [page 533]
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(short) Method [page 534]
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.

1.1.37.11.1 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 ()
```

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

The LookupBegin method must be called before each search.

Related Information

LookupBegin() Method [page 532]
LookupForward(short) Method [page 534]
IsEOF Property [page 307]

1.1.37.11.2 LookupForward(short) 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 unsafe 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.

Returns

True if successful, false otherwise.

Exceptions

ULException class A SQL error occurred.

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 (as shown by the ULDataReader.IsEOF property).

The LookupBegin method must be called before each search.

Related Information

LookupBegin() Method [page 532]
LookupForward() Method [page 533]
IsEOF Property [page 307]
LookupBegin() Method [page 532]

1.1.37.12 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.

Related Information

DeleteAllRows() Method [page 514]

1.1.37.13 Schema Property

Holds the table schema.

Syntax

Visual Basic

```
Public ReadOnly Shadows Property Schema As ULTableSchema
```

C#

```
public new ULTableSchema Schema {get;}
```

Remarks

This property is only valid while its connection is open.

The ULTableSchema object representing the table schema.

This property represents the complete schema of the table, including UltraLite.NET extended information which is not represented in the results from calling the ULDataReader.GetSchemaTable method.

Related Information

ULTableSchema Class [page 537]

1.1.38 ULTableSchema Class

UL Ext: Represents the schema of an UltraLite table.

Syntax

Visual Basic

```
Public NotInheritable Class ULTableSchema Inherits ULCursorSchema
```

C#

```
public sealed class ULTableSchema : ULCursorSchema
```

Members

All members of ULTableSchema, including inherited members.

Methods

Modifier and Type	Method	Description
public unsafe string	GetColumnDefaultValue(int) [page 541]	Returns the default value of the specified column.
public unsafe ulong	GetColumnPartitionSize(int) [page 542]	Returns the global autoincrement partition size assigned to the specified column.
public unsafe ULIndexSchema	GetIndex(string) [page 543]	Returns the index schema of the named index.
public unsafe string	GetIndexName(int) [page 543]	Returns the name of the index identified by the specified index ID.
public unsafe string	GetOptimalIndex(int) [page 544]	The optimal index for searching a table using the specified column.
public unsafe string	GetPublicationPredicate(string) [page 545]	Returns the publication predicate for this table in the named publication.
public unsafe bool	IsColumnAutoIncrement(int) [page 546]	Checks whether the specified column's default is set to autoincrement.
public unsafe bool	IsColumnCurrentDate(int) [page 547]	Checks whether the specified column's default is set to the current date (a ULDbType.Date value).
public unsafe bool	IsColumnCurrentTime(int) [page 548]	Checks whether the specified column's default is set to the current time (a ULDbType.Time value).
public unsafe bool	IsColumnCurrentTimestamp(int) [page 549]	Checks whether the specified column's default is set to the current timestamp (a ULDbType.TimeStamp value).

Modifier and Type	Method	Description
public unsafe bool	IsColumnCurrentUTCTimestamp(int) [page 550]	Checks whether the specified column's default is set to the current UTC timestamp (a ULDbType.TimeStamp value).
public unsafe bool	IsColumnGlobalAutoIncrement(int) [page 551]	Checks whether the specified column's default is set to global autoincrement.
public unsafe bool	IsColumnNewUUID(int) [page 552]	Checks whether the specified column's default is set to a new UUID (a System.Guid value).
public unsafe bool	IsColumnNullable(int) [page 553]	Checks whether the specified column is nullable.
public unsafe bool	IsInPublication(string) [page 554]	Checks whether the table is contained in the named publication.
protected virtual override void	VerifyOpen() [page 554]	

Properties

Modifier and Type	Property	Description
public unsafe int	IndexCount [page 555]	Returns the number of indexes on the table.
public unsafe bool	IsNeverSynchronized [page 555]	Checks whether the table is marked as never being synchronized.
public override string	Name [page 556]	Returns the name of the table.
public unsafe ULIndexSchema	PrimaryKey [page 556]	Returns the index schema of the primary key for the table.
public unsafe bool	UploadUnchangedRows [page 557]	Checks whether the database uploads rows that have not changed.

Inherited members from ULCursorSchema

Modifier and Type	Member	Description
public short	ColumnCount [page 223]	Returns the number of columns in the cursor.
protected unsafe void	GetColumnCount() [page 215]	
public unsafe short	GetColumnID(string) [page 215]	Returns the column ID of the named column.
public string	GetColumnName(int) [page 216]	Returns the name of the column identified by the specified column ID.
public unsafe int	GetColumnPrecision(int) [page 217]	Returns the precision of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).

Modifier and Type	Member	Description
public unsafe int	GetColumnScale(int) [page 218]	Returns the scale of the column identified by the specified column ID if the column is a numeric column (the NUMERIC SQL type).
public unsafe int	GetColumnSize(int) [page 219]	Returns the size of the column identified by the specified column ID if the column is a sized column (the BINARY or CHAR SQL types).
public string	GetColumnSQLName(int) [page 220]	Returns the name of the column identified by the specified column ID.
public unsafe ULDbType	GetColumnULDbType(int) [page 221]	Returns the UltraLite.NET data type of the column identified by the specified column ID.
public unsafe DataTable	GetSchemaTable() [page 222]	Returns a System.Data.DataTable that describes the column schema of the ULDataReader object.
public bool	IsOpen [page 223]	Checks whether the cursor schema is currently open.

Remarks

There is no constructor for this class. A ULTableSchema object is attached to a table as its ULTable.Schema property.

In this section:

GetColumnDefaultValue(int) Method [page 541]
Returns the default value of the specified column.

GetColumnPartitionSize(int) Method [page 542]
Returns the global autoincrement partition size assigned to the specified column.

GetIndex(string) Method [page 543]
Returns the index schema of the named index.

GetIndexName(int) Method [page 543]
Returns the name of the index identified by the specified index ID.

GetOptimalIndex(int) Method [page 544]
The optimal index for searching a table using the specified column.

GetPublicationPredicate(string) Method [page 545]
Returns the publication predicate for this table in the named publication.

IsColumnAutoIncrement(int) Method [page 546]
Checks whether the specified column's default is set to autoincrement.

IsColumnCurrentDate(int) Method [page 547]
Checks whether the specified column's default is set to the current date (a ULDbType.Date value).

IsColumnCurrentTime(int) Method [page 548]
Checks whether the specified column's default is set to the current time (a ULDbType.Time value).

IsColumnCurrentTimestamp(int) Method [page 549]
Checks whether the specified column's default is set to the current timestamp (a ULDbType.TimeStamp value).

IsColumnCurrentUTCTimestamp(int) Method [page 550]
Checks whether the specified column's default is set to the current UTC timestamp (a ULDbType.TimeStamp value).

IsColumnGlobalAutoIncrement(int) Method [page 551]
Checks whether the specified column's default is set to global autoincrement.

IsColumnNewUUID(int) Method [page 552]
Checks whether the specified column's default is set to a new UUID (a System.Guid value).

IsColumnNullable(int) Method [page 553]
Checks whether the specified column is nullable.

IsInPublication(string) Method [page 554]
Checks whether the table is contained in the named publication.

VerifyOpen() Method [page 554]

IndexCount Property [page 555]
Returns the number of indexes on the table.

IsNeverSynchronized Property [page 555]
Checks whether the table is marked as never being synchronized.

Name Property [page 556]
Returns the name of the table.

PrimaryKey Property [page 556]
Returns the index schema of the primary key for the table.

UploadUnchangedRows Property [page 557]
Checks whether the database uploads rows that have not changed.

Related Information

Schema Property [page 536]
ULCursorSchema Class [page 212]

1.1.38.1 GetColumnDefaultValue(int) Method

Returns the default value of the specified column.

Syntax

Visual Basic

```
Public Function GetColumnDefaultValue (ByVal columnID As Integer) As String
```

C#

```
public unsafe 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.

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.38.2 GetColumnPartitionSize(int) Method

Returns the global autoincrement partition size assigned to the specified column.

Syntax

Visual Basic

```
Public Function GetColumnPartitionSize (ByVal columnID As Integer) As ULong
```

C#

```
public unsafe 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.

Returns

The column's global autoincrement partition size as a System.UInt64 structure.

Exceptions

ULException class A SQL error occurred.

Remarks

All global autoincrement columns in a given table share the same global autoincrement partition.

Related Information

IsColumnGlobalAutoIncrement(int) Method [page 551]
ColumnCount Property [page 223]

1.1.38.3 GetIndex(string) Method

Returns the index schema of the named index.

Syntax

Visual Basic

```
Public Function GetIndex (ByVal name As String) As ULIndexSchema
```

C#

```
public unsafe ULIndexSchema GetIndex (string name)
```

Parameters

name The name of the index.

Returns

A ULIndexSchema object representing the named index.

Exceptions

ULException class A SQL error occurred.

Related Information

ULIndexSchema Class [page 346]

1.1.38.4 GetIndexName(int) 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 unsafe string GetIndexName (int indexID)
```

Parameters

indexID The ID of the index. The value must be in the range [1,IndexCount].

Returns

The name of the index as a string.

Exceptions

ULException class A SQL error occurred.

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.

Related Information

IndexCount Property [page 555]

1.1.38.5 GetOptimalIndex(int) Method

The optimal index for searching a table using the specified column.

Syntax

Visual Basic

```
Public Function GetOptimalIndex (ByVal columnID As Integer) As String
```

C#

```
public unsafe string GetOptimalIndex (int columnID)
```

Parameters

columnID The ID number of the column. The first column in the table has an ID value of zero.

Returns

A ULIndexSchema object representing the optimal index for the specified column.

Exceptions

ULException class A SQL error occurred.

Remarks

The specified column is the first column in the index, but the index may have more than one column.

Related Information

ColumnCount Property [page 223]
ULIndexSchema Class [page 346]

1.1.38.6 GetPublicationPredicate(string) 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 unsafe string GetPublicationPredicate (string pubName)
```

Parameters

pubName The name of the publication.

Returns

The publication predicate as a string.

Exceptions

ULException class A SQL error occurred.

1.1.38.7 IsColumnAutoIncrement(int) 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 unsafe 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.

Returns

True if the column is autoincrementing, false if it is not autoincrementing.

Exceptions

ULException class A SQL error occurred.

Related Information

ColumnCount Property [page 223]

1.1.38.8 IsColumnCurrentDate(int) Method

Checks whether the specified column's default is set to the current date (a ULDbType.Date value).

Syntax

Visual Basic

```
Public Function IsColumnCurrentDate (ByVal columnID As Integer) As Boolean
```

C#

```
public unsafe 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.

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.38.9 IsColumnCurrentTime(int) Method

Checks whether the specified column's default is set to the current time (a ULDbType.Time value).

Syntax

Visual Basic

```
Public Function IsColumnCurrentTime (ByVal columnID As Integer) As Boolean
```

C#

```
public unsafe 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.

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.38.10 IsColumnCurrentTimestamp(int) Method

Checks whether the specified column's default is set to the current timestamp (a ULDbType.TimeStamp value).

Syntax

Visual Basic

```
Public Function IsColumnCurrentTimestamp (ByVal columnID As Integer) As
Boolean
```

C#

```
public unsafe 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.

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.38.11 IsColumnCurrentUTCTimestamp(int) Method

Checks whether the specified column's default is set to the current UTC timestamp (a ULDbType.TimeStamp value).

Syntax

Visual Basic

```
Public Function IsColumnCurrentUTCTimestamp (ByVal columnID As Integer) As
Boolean
```

C#

```
public unsafe bool IsColumnCurrentUTCTimestamp (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.

Returns

True if the column defaults to the current UTC timestamp, false if the column does not default to the current UTC timestamp.

Exceptions

ULException class A SQL error occurred.

Related Information

ColumnCount Property [page 223]

1.1.38.12 IsColumnGlobalAutoIncrement(int) 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 unsafe 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.

Returns

True if the column is global autoincrementing, false if it is not global autoincrementing.

Exceptions

ULException class A SQL error occurred.

Related Information

GetColumnPartitionSize(int) Method [page 542]
DatabaseID Property [page 165]
ColumnCount Property [page 223]

1.1.38.13 IsColumnNewUUID(int) Method

Checks whether the specified column's default is set to a new UUID (a System.Guid value).

Syntax

Visual Basic

```
Public Function IsColumnNewUUID (ByVal columnID As Integer) As Boolean
```

C#

```
public unsafe 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.

Returns

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.

Related Information

ColumnCount Property [page 223]

1.1.38.14 IsColumnNullable(int) Method

Checks whether the specified column is nullable.

Syntax

Visual Basic

```
Public Function IsColumnNullable (ByVal columnID As Integer) As Boolean
```

C#

```
public unsafe 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.

Returns

True if the column is nullable, false if it is not nullable.

Exceptions

ULException class A SQL error occurred.

Related Information

ColumnCount Property [page 223]

1.1.38.15 IsInPublication(string) 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 unsafe bool IsInPublication (string pubName)
```

Parameters

pubName The name of the publication.

Returns

True if the table is in the publication, false if the table is not in the publication.

Exceptions

ULException class A SQL error occurred.

1.1.38.16 VerifyOpen() Method

Syntax

Visual Basic

```
Protected Overridable Overrides Sub VerifyOpen ()
```

C#

```
protected virtual override void VerifyOpen ()
```

1.1.38.17 IndexCount Property

Returns the number of indexes on the table.

Syntax

Visual Basic

```
Public ReadOnly Property IndexCount As Integer
```

C#

```
public unsafe int IndexCount {get;}
```

Remarks

The number of indexes on the table or 0 if the table schema is closed.

Index IDs range from 1 to the IndexCount value, 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.

1.1.38.18 IsNeverSynchronized Property

Checks whether the table is marked as never being synchronized.

Syntax

Visual Basic

```
Public ReadOnly Property IsNeverSynchronized As Boolean
```

C#

```
public unsafe bool IsNeverSynchronized {get;}
```

Remarks

True if the table is marked as never being synchronized, false otherwise.

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.

1.1.38.19 Name Property

Returns the name of the table.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property Name As String
```

C#

```
public override string Name {get;}
```

Remarks

The name of the table as a string.

1.1.38.20 PrimaryKey Property

Returns the index schema of the primary key for the table.

Syntax

Visual Basic

```
Public ReadOnly Property PrimaryKey As ULIndexSchema
```

C#

```
public unsafe ULIndexSchema PrimaryKey {get;}
```

Remarks

A ULIndexSchema object representing the primary key for the table.

Related Information

ULIndexSchema Class [page 346]

1.1.38.21 UploadUnchangedRows Property

Checks whether the database uploads rows that have not changed.

Syntax

Visual Basic

```
Public ReadOnly Property UploadUnchangedRows As Boolean
```

C#

```
public unsafe bool UploadUnchangedRows {get;}
```

Remarks

True if the table is marked to always upload all rows during synchronization, false if the table is marked to upload only changed rows.

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.

1.1.39 ULTransaction Class

Represents a SQL transaction.

Syntax

Visual Basic

```
Public NotInheritable Class ULTransaction Inherits
System.Data.Common.DbTransaction
```

C#

```
public sealed class ULTransaction : System.Data.Common.DbTransaction
```

Members

All members of ULTransaction, including inherited members.

Methods

Modifier and Type	Method	Description
public override void	Commit() [page 559]	Commits the database transaction.
protected override void	Dispose(bool) [page 559]	
public override void	Rollback() [page 560]	Rolls back the transaction's outstanding changes to the database.

Properties

Modifier and Type	Property	Description
public new ULConnection	Connection [page 560]	Returns the connection associated with the transaction.
protected override DbConnection	DbConnection [page 561]	
public override IsolationLevel	IsolationLevel [page 561]	Returns the isolation level for the transaction.

Remarks

There is no constructor for the ULTransaction class. To obtain a ULTransaction object, use the ULConnection.BeginTransaction method. To associate a command with a transaction, use the ULCommand.Transaction property.

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.

In this section:

Commit() Method [page 559]
: Commits the database transaction.

Dispose(bool) Method [page 559]

Rollback() Method [page 560]
: Rolls back the transaction's outstanding changes to the database.

Connection Property [page 560]
: Returns the connection associated with the transaction.

DbConnection Property [page 561]

IsolationLevel Property [page 561]
: Returns the isolation level for the transaction.

Related Information

BeginTransaction() Method [page 121]
Transaction Property [page 91]

1.1.39.1 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 a Commit method call 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 method to complete the transaction.

Related Information

Rollback() Method [page 560]

1.1.39.2 Dispose(bool) Method

Syntax

Visual Basic

```
Protected Overrides Sub Dispose (ByVal disposing As Boolean)
```

C#

```
protected override void Dispose (bool disposing)
```

1.1.39.3 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.

Related Information

Commit() Method [page 559]

1.1.39.4 Connection Property

Returns the connection associated with the transaction.

Syntax

Visual Basic

```
Public ReadOnly Shadows Property Connection As ULConnection
```

C#

```
public new ULConnection Connection {get;}
```

Remarks

The ULConnection object associated with the transaction, or a null reference (Nothing in Visual Basic) if the transaction is no longer valid.

This is the strongly-typed version of the System.Data.IDbTransaction.Connection and System.Data.Common.DbCommand.Connection properties.

Related Information

BeginTransaction() Method [page 121]
ULConnection Class [page 108]

1.1.39.5 DbConnection Property

Syntax

Visual Basic

```
Protected ReadOnly Overrides Property DbConnection As DbConnection
```

C#

```
protected override DbConnection DbConnection {get;}
```

1.1.39.6 IsolationLevel Property

Returns the isolation level for the transaction.

Syntax

Visual Basic

```
Public ReadOnly Overrides Property IsolationLevel As IsolationLevel
```

C#

```
public override IsolationLevel IsolationLevel {get;}
```

Remarks

One of the System.Data.IsolationLevel values. UltraLite.NET only supports the System.Data.IsolationLevel.ReadUncommitted value.

Related Information

BeginTransaction() Method [page 121]

1.1.40 ULInfoMessageEventHandler(object, ULInfoMessageEventArgs) Delegate

Represents the method that handles the ULConnection.InfoMessage event.

Syntax

Visual Basic

```
Public Delegate Sub ULInfoMessageEventHandler (
    ByVal obj As Object,
    ByVal args As ULInfoMessageEventArgs
) As delegate void
```

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.

Related Information

InfoMessage Event [page 170]

ULInfoMessageEventArgs Class [page 356]

1.1.41 ULRowsCopiedEventHandler(object, ULRowsCopiedEventArgs) Delegate

Represents the method that handles the ULBulkCopy.ULRowsCopied event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowsCopiedEventHandler (
    ByVal sender As Object,
    ByVal rowsCopiedEventArgs As ULRowsCopiedEventArgs
) As delegate void
```

C#

```
public delegate void ULRowsCopiedEventHandler (
    object sender,
    ULRowsCopiedEventArgs rowsCopiedEventArgs
);
```

Remarks

The ULRowsCopiedEventHandler delegate is not available in the .NET Compact Framework 2.0.

Related Information

ULRowsCopied Event [page 24]

1.1.42 ULRowUpdatedEventHandler(object, ULRowUpdatedEventArgs) Delegate

Represents the method that handles the ULDataAdapter.RowUpdated event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatedEventHandler (
    ByVal sender As Object,
    ByVal e As ULRowUpdatedEventArgs
) As delegate void
```

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.

Related Information

RowUpdated Event [page 238]
ULRowUpdatedEventArgs Class [page 458]

1.1.43 ULRowUpdatingEventHandler(object, ULRowUpdatingEventArgs) Delegate

Represents the method that handles the ULDataAdapter.RowUpdating event.

Syntax

Visual Basic

```
Public Delegate Sub ULRowUpdatingEventHandler (
    ByVal sender As Object,
    ByVal e As ULRowUpdatingEventArgs
) As delegate void
```

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.

Related Information

RowUpdating Event [page 239]
ULRowUpdatingEventArgs Class [page 461]

1.1.44 ULSyncProgressedDlg(IAsyncResult, ULSyncProgressData) Delegate

Represents the method that is invoked during synchronization with synchronization progress information.

Syntax

Visual Basic

```
Public Delegate Sub ULSyncProgressedDlg (
    ByVal result As IAsyncResult,
    ByVal data As ULSyncProgressData
) As delegate void
```

C#

```
public delegate void ULSyncProgressedDlg (
    IAsyncResult result,
    ULSyncProgressData data
);
```

Parameters

result The IAsyncResult object returned from the BeginSynchronize method. Use result.AsyncState to access the object provided to the BeginSynchronize method.

data A ULSyncProgressData object containing the latest synchronization progress data.

Remarks

It is safe to do GUI work or to make UltraLite.NET API calls in this method. The synchronization is not being held up during calls to this method.

Related Information

BeginSynchronize(Control, ULSyncProgressedDlg, object) Method [page 119]
ULSyncProgressData Class [page 484]

1.1.45 ULAuthStatusCode Enumeration

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

Syntax

Visual Basic

```
Public Enum ULAuthStatusCode
```

C#

```
enum ULAuthStatusCode
```

Members

Member name	Description	Value
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 expire soon (VALID_BUT_EXPIRES_SOON = 2).	2
EXPIRED	User ID or password has expired - authorization failed (EXPIRED = 3).	3
INVALID	Bad user ID or password - authorization failed (INVALID = 4).	4
IN_USE	User ID is already in use - authorization failed (IN_USE = 5).	5

Related Information

AuthStatus Property [page 503]

1.1.46 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#

```
enum ULBulkCopyOptions
```

Members

Member name	Description	Value
Default	Specifying only this causes the default behavior to be used.	0x0
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.	0x1
UseInternalTransaction	When specified, each batch of the bulk-copy operation is executed within a transaction. When not specified, transactions aren't used. If you indicate this option and also provide a ULTransaction object to the constructor, a System.ArgumentException occurs.	0x2

Remarks

The ULBulkCopyOptions class is not available in the .NET Compact Framework 2.0.

The ULBulkCopyOptions enumeration is used when you construct a ULBulkCopy instance to specify how WriteToServer methods behave.

Related Information

ULBulkCopy Class [page 11]

1.1.47 ULDateOrder Enumeration

UL Ext: Enumerates the date orders that a database can support.

Syntax

Visual Basic

```
Public Enum ULDateOrder
```

C#

```
enum ULDateOrder
```

Members

Member name	Description
YMD	The year followed by month, followed by day of the month.
MDY	The month followed by day of the month, followed by year.
DMY	The day of the month followed by month, followed by year.

1.1.48 ULDbType Enumeration

Enumerates the UltraLite.NET database data types.

Syntax

Visual Basic

```
Public Enum ULDbType
```

C#

```
enum ULDbType
```

Members

Member name	Description	Value
BigInt	Signed 64-bit integer.	ULNET_TYPE_S_BIG
Binary	Binary data, with a specified maximum length. The *Binary* and *VarBinary* enumeration values are aliases of each other.	ULNET_TYPE_BINARY
Bit	1-bit flag.	ULNET_TYPE_BIT
Char	Character data, with a specified length. In UltraLite.NET, this type always supports Unicode characters. The *Char* and *VarChar* types are fully compatible.	ULNET_TYPE_FIXCHAR
Date	Date information.	ULNET_TYPE_DATE
DateTime	Timestamp information (date, time). The *DateTime* and *TimeStamp* enumeration values are aliases of each other.	ULNET_TYPE_TIMESTAMP
Decimal	Exact numerical data, with a specified precision and scale. The *Decimal* and *Numeric* enumeration values are aliases of each other.	ULNET_TYPE_NUMERIC
Double	Double precision floating-point number (8 bytes).	ULNET_TYPE_DOUBLE
Float	Single precision floating-point number (4 bytes). The *Float* and *Real* enumeration values are aliases of each other.	ULNET_TYPE_REAL
Integer	Unsigned 32-bit integer.	ULNET_TYPE_S_LONG
LongBinary	Binary data, with variable length.	ULNET_TYPE_LONGBINARY
LongVarchar	Character data, with variable length. In UltraLite.NET, this type always supports Unicode characters.	ULNET_TYPE_LONGVARCHAR
Numeric	Exact numerical data, with a specified precision and scale. The *Decimal* and *Numeric* enumeration values are aliases of each other.	ULNET_TYPE_NUMERIC

Member name	Description	Value
Real	Single precision floating-point number (4 bytes). The *Float* and *Real* enumeration values are aliases of each other.	ULNET_TYPE_REAL
SmallInt	Signed 16-bit integer.	ULNET_TYPE_S_SHORT
STGeometry	ST_Geometry information.	ULNET_TYPE_ST_GEOMETRY
Time	Time information.	ULNET_TYPE_TIME
TimeStamp	Timestamp information (date, time). The *DateTime* and *TimeStamp* enumeration values are aliases of each other.	ULNET_TYPE_TIMESTAMP
TimeStampWithTimeZone	Timestamp information (date, time) along with the time zone offset.	ULNET_TYPE_TIMESTAMP_WITH_TIME_ZONE
TinyInt	Unsigned 8-bit integer.	ULNET_TYPE_TINY
UniqueIdentifier	Universally Unique Identifier (UUID/GUID).	ULNET_TYPE_UUID
UnsignedBigInt	Unsigned 64-bit integer.	ULNET_TYPE_U_BIG
UnsignedInt	Unsigned 32-bit integer.	ULNET_TYPE_U_LONG
UnsignedSmallInt	Unsigned 16-bit integer.	ULNET_TYPE_U_SHORT
VarBinary	Binary data, with a specified maximum length. The *Binary* and *VarBinary* enumeration values are aliases of each other.	ULNET_TYPE_BINARY
VarChar	Character data, with a specified maximum length. In UltraLite.NET, this type always supports Unicode characters. The *Char* and *VarChar* types are fully compatible.	ULNET_TYPE_CHAR

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

ULDbType	Compatible .NET type	C# built-in type	Visual Basic built-in type
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.Decimal	decimal	Decimal
Float, *Real*	System.Single	float	Single
BigInt	System.Int64	long	Long
Integer	System.Int32	int	Integer
SmallInt	System.Int16	short	Short
ST_Geometry	System.String	String	String
Time	System.TimeSpan	TimeSpan (no built-in type)	TimeSpan (no built-in type)
DateTime, *TimeStamp*	System.DateTime	DateTime (no built-in type)	Date
TimeStampWithTimeZone	System.String	String	String
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.

Related Information

GetFieldType(int) Method [page 279]
GetDataTypeName(int) Method [page 274]
GetColumnULDbType(int) Method [page 221]

1.1.49 ULDBValid Enumeration

Enumerates the UltraLite.NET database validation methods

Syntax

Visual Basic

```
Public Enum ULDBValid
```

C#

```
enum ULDBValid
```

Members

Member name	Description	Value
EXPRESS_VALIDATE	Performs a faster, though less thorough, validation.	0
FULL_VALIDATE	Validates tables, indexes, and all database pages.	1

Related Information

ValidateDatabase(string, ULDBValid) Method [page 247]
ValidateDatabase(ULDBValid) Method [page 161]
ValidateDatabase(ULDBValid, string) Method [page 162]

1.1.50 ULRuntimeType Enumeration

UL Ext: Enumerates the types of UltraLite.NET runtimes.

Syntax

Visual Basic

```
Public Enum ULRuntimeType
```

C#

```
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. Databases can be shared by different applications.	1

Related Information

RuntimeType Property [page 249]

1.1.51 ULSqlProgressState Enumeration

UL Ext: Enumerates all the states that can occur while executing SQL passthrough scripts.

Syntax

Visual Basic

```
Public Enum ULSqlProgressState
```

C#

```
enum ULSqlProgressState
```

Members

Member name	Description	Value
STATE_STARTING	No scripts have been executed yet.	0
STATE_RUNNING_SCRIPT	Currently running a SQL passthrough script.	1

Member name	Description	Value
STATE_DONE	Scripts have successfully completed.	2
STATE_ERROR	Scripts have completed, but an error occurred.	3

Related Information

ULSqlProgressData Class (Deprecated) [page 466]

1.1.52 ULStreamType Enumeration

UL Ext: Enumerates the types of MobiLink synchronization streams to use for synchronization.

Syntax

Visual Basic

```
Public Enum ULStreamType
```

C#

```
enum ULStreamType
```

Members

Member name	Description
TCPIP	Synchronize via TCP/IP.
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).
TLS	Synchronize via TCP/IP with transport layer security.

Related Information

Stream Property [page 481]

1.1.53 ULSyncProgressState Enumeration

UL Ext: Enumerates all the states that can occur while synchronizing.

Syntax

Visual Basic

```
Public Enum ULSyncProgressState
```

C#

```
enum ULSyncProgressState
```

Members

Member name	Description	Value
STATE_STARTING	No synchronization actions have been taken yet.	0
STATE_CONNECTING	The synchronization stream has been built, but is not yet opened.	1
STATE_RESUMING_DOWNLOAD	An optional state that is entered when we are attempting to resume a partial download. On success, sync will proceed to the STATE_RECEIVING_TABLE state, and STATE_ERROR if we are unable to resume.	2
STATE_SENDING_HEADER	The synchronization stream has been opened and the header is about to be sent.	3
STATE_SENDING_CHECK_SYNC_REQUEST	The state of the last upload is unknown, so a request to check its status is being sent.	4
STATE_WAITING_FOR_CHECK_SYNC_RESPONSE	Waiting for the server to respond to the check sync request.	5

Member name	Description	Value
STATE_PROCESS-ING_CHECK_SYNC_RESPONSE	The response to the check sync request has been received and is being processed.	6
STATE_SENDING_TABLE	A table is being sent. Progress can be monitored using the ULSyncProgressData.SyncTableIndex and ULSyncProgressData.SyncTable-Count properties.	7
STATE_SENDING_DATA	Data for the current table is being sent. The ULSyncProgressData.SentBytes, ULSyncProgressData.SentInserts, UL-SyncProgressData.SentUpdates, and ULSyncProgressData.SentDeletes properties have been updated.	8
STATE_FINISHING_UPLOAD	The upload is completing. The final count of rows sent is included with this event.	9
STATE_WAITING_FOR_UPLOAD_ACK	Waiting for the server to acknowledge receiving our upload.	10
STATE_PROCESSING_UPLOAD_ACK	The server has acknowledged receiving our upload.	11
STATE_WAITING_FOR_DOWNLOAD	Waiting for the server to start sending the download.	12
STATE_RECEIVING_TABLE	A table is being received. Progress can be monitored using the ULSyncProgressData.SyncTableIndex and ULSyncProgressData.SyncTable-Count properties.	13
STATE_RECEIVING_DATA	Data for the current table is being received. The ULSyncProgressData.Received-Bytes, ULSyncProgressData.Receive-dInserts, ULSyncProgressData.Receive-dUpdates, and ULSyncProgress-Data.ReceivedDeletes properties have been updated.	14
STATE_COMMITTING_DOWNLOAD	The download is being committed. The final count of rows received is included with this event.	15

Member name	Description	Value
STATE_ROLLING_BACK_DOWNLOAD	Synchronization is rolling back the download because an error was encountered during the download. The error is reported with a subsequent STATE_ERROR progress report.	16
STATE_SENDING_DOWNLOAD_ACK	An acknowledgement of download completion is being sent.	17
STATE_DISCONNECTING	The synchronization stream is about to be closed.	18
STATE_DONE	Synchronization has successfully completed.	19
STATE_ERROR	Synchronization has completed, but an error occurred.	20
STATE_CANCELLED	Synchronization has been canceled.	21

Related Information

ULSyncProgressData Class [page 484]
ReceivedBytes Property [page 490]
ReceivedInserts Property [page 491]
ReceivedUpdates Property [page 492]
ReceivedDeletes Property [page 491]
SentBytes Property [page 493]
SentInserts Property [page 494]
SentUpdates Property [page 494]
SentDeletes Property [page 493]
SyncTableIndex Property [page 496]
SyncTableCount Property [page 496]

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP