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

MobiLink - .NET API Reference

THE BEST RUN SAP

Content

1 MobiLink Server .NET API Reference

The MobiLink .NET API topics explain interfaces and classes, and their associated methods, properties, and constructors.

Namespace

```
Sap.MobiLink.Script
```

Remarks

To use these classes, reference the `Sap.MobiLink.Script.dll` assembly, located in *%SQLANY17%* `\Assembly\V3.5`.

The MobiLink server .NET API reference is available in the *MobiLink - .NET API Reference* at https://help.sap.com/viewer/935ac449a8764285843c9cb0012d5f05/LATEST/en-US.

In this section:

Signals MobiLink that a fatal server-side error has occurred internally and should shutdown immediately.

LogMessage Class [page 69]
Contains information regarding a message printed to the log.

NetworkData Interface [page 73]
Contains information about the network streams for a synchronization.

ScriptExecutionException Class [page 79]
Signals that an error has occurred in a user script.

ServerContext Interface [page 83]
Instantiates the context that is present for the duration of the MobiLink server.

ServerException Class [page 90]
Sends a signal to MobiLink when an error has occurred with the server to indicate that it should shut down immediately.

SpatialUtilities Class [page 94]
Represents a collection of static methods to work with spatial values.

SynchronizationException Class [page 97]
Indicates when a synchronization exception has occurred and that the current synchronization should be rolled back and restarted.

UpdateDataReader Interface [page 101]
Holds the update operations for one upload transaction for one table.

UploadData Interface [page 103]
Encapsulates upload operations for direct row handling.

UploadedTableData Interface [page 106]
Encapsulates information for one uploaded table for a synchronization.

LogCallback(ServerContext, LogMessage) Delegate [page 112]
Called when the MobiLink server prints a message.

ShutdownCallback(ServerContext) Delegate [page 113]
Called when MobiLink server is shutting down.

SQLType Enumeration [page 113]
Enumerates all possible ODBC data types.

1.1 DateTimeWithTimeZone Class

Represents a DateTime with time zone offsets.

Syntax

Visual Basic

```
Public NotInheritable Class DateTimeWithTimeZone
```

C#

```
public sealed class DateTimeWithTimeZone
```

Members

All members of DateTimeWithTimeZone, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	DateTimeWithTimeZone [page 10]	Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with time zone offsets of 0.

Methods

Modifier and Type	Method	Description
public static DateTimeWithTimeZone	Parse(string) [page 13]	Parses the given string and returns a new DateTimeWithTimeZone.
public override string	ToString [page 14]	The equivalent to DateTime.ToString() with the time zone offset appended to the end.

Properties

Modifier and Type	Property	Description
public DateTime	DateTime [page 16]	Gets the DateTime that corresponds to this without the time zone offset.
public int	Day [page 16]	The equivalent to DateTime.Day.
public int	Hour [page 16]	The equivalent to DateTime.Hour.
public int	Millisecond [page 17]	The equivalent to DateTime.Millisecond.
public int	Minute [page 17]	The equivalent to DateTime.Minute.
public int	Month [page 17]	The equivalent to DateTime.Month.
public int	Second [page 18]	The equivalent to DateTime.Second.
public int	TimeZoneHour [page 18]	Gets the hours part of the time zone offset.
public int	TimeZoneMinute [page 18]	Gets the minutes part of the time zone offset.
public int	Year [page 19]	The equivalent to DateTime.Year.

In this section:

1.1.1 DateTimeWithTimeZone Constructor

Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with time zone offsets of 0.

Overload list

Modifier and Type	Overload name	Description
public	DateTimeWithTimeZone(DateTime) [page 10]	Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with time zone offsets of 0.
public	DateTimeWithTimeZone(DateTime, int, int) [page 11]	Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with the provided time zone offset.
public	DateTimeWithTimeZone(int, int, int, int, int, int, int, int, int, int) [page 12]	Constructs a DateTimeWithTimeZone with the specified year, month, day, hour, minute, second, millisecond, time zone hour and time zone minute.

In this section:

DateTimeWithTimeZone(DateTime) Constructor [page 10]
Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with time zone offsets of 0.

DateTimeWithTimeZone(DateTime, int, int) Constructor [page 11]
Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with the provided time zone offset.

DateTimeWithTimeZone(int, int, int, int, int, int, int, int, int) Constructor [page 12]
Constructs a DateTimeWithTimeZone with the specified year, month, day, hour, minute, second, millisecond, time zone hour and time zone minute.

1.1.1.1 DateTimeWithTimeZone(DateTime) Constructor

Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with time zone offsets of 0.

Syntax

Microsoft Visual Basic

```
Public Sub DateTimeWithTimeZone (ByVal dt As Date)
```

C#

```
public DateTimeWithTimeZone (DateTime dt)
```

1.1.1.2 DateTimeWithTimeZone(DateTime, int, int) Constructor

Constructs a DateTimeWithTimeZone with the same date and time as the specified DateTime with the provided time zone offset.

Syntax

Microsoft Visual Basic

```
Public Sub DateTimeWithTimeZone (
    ByVal dt As Date,
    ByVal tz_hour As Integer,
    ByVal tz_minute As Integer
)
```

C#

```
public DateTimeWithTimeZone (
    DateTime dt,
    int tz_hour,
    int tz_minute
)
```

Parameters

dt The date and time portions.

tz_hour The time zone offset hours. (-12 through 14)

tz_minute The time zone offset minutes. (-59 through 59). This value can be negative only if *tz_hour* is not positive.

Exceptions

System.ArgumentOutOfRangeException Thrown when tz_hour or tz_minute are out of range.

1.1.1.3 DateTimeWithTimeZone(int, int, int, int, int, int, int, int) Constructor

Constructs a DateTimeWithTimeZone with the specified year, month, day, hour, minute, second, millisecond, time zone hour and time zone minute.

Syntax

Microsoft Visual Basic

```
Public Sub DateTimeWithTimeZone (
    ByVal year As Integer,
    ByVal month As Integer,
    ByVal day As Integer,
    ByVal hour As Integer,
    ByVal minute As Integer,
    ByVal second As Integer,
    ByVal millisecond As Integer,
    ByVal tz_hour As Integer,
    ByVal tz_minute As Integer
)
```

C#

```
public DateTimeWithTimeZone (
    int year,
    int month,
    int day,
    int hour,
    int minute,
    int second,
    int millisecond,
    int tz_hour,
    int tz_minute
)
```

Parameters

year The year. (1 through 9999)
month The month. (1 through 12)
day The day. (1 through the number of days in *month*)
hour The hours. (0 through 23)
minute The minutes. (0 through 59)
second The seconds. (0 through 59)
millisecond The milliseconds. (1 through 999)
tz_hour The time zone offset hours. (-12 through 14)
tz_minute The time zone offset minutes. (-59 through 59). This value can be negative only if *tz_hour* is not positive.

Exceptions

System.ArgumentOutOfRangeException Thrown when any parameter is out of range.

1.1.2 Parse(string) Method

Parses the given string and returns a new DateTimeWithTimeZone.

Syntax

Visual Basic

```
Public Shared Function Parse (ByVal val As String) As DateTimeWithTimeZone
```

C#

```
public static DateTimeWithTimeZone Parse (string val)
```

Parameters

val A string of the form "yyyy-MM-dd HH:mm:ss.ffffff SHH:mm", where S is the sign on the time zone hours offset and the second HH:mm is the time zone offset. The fractional part of the time can be omitted. The time zone offset can be omitted. If the time zone offset is present, the sign can be omitted.

Exceptions

System.FormatException Thrown when the given string does not match that format.

1.1.3 ToString Method

The equivalent to DateTime.ToString() with the time zone offset appended to the end.

Overload list

Modifier and Type	Overload name	Description
public override string	ToString() [page 14]	The equivalent to DateTime.ToString() with the time zone offset appended to the end.
public string	ToString(IFormatProvider) [page 15]	The equivalent to DateTime.ToString(provider) with the time zone offset appended to the end.
public string	ToString(string) [page 15]	The equivalent to DateTime.ToString(format) with the time zone offset appended to the end.
public string	ToString(string, IFormatProvider) [page 15]	The equivalent to DateTime.ToString(format, provider) with the time zone offset appended to the end.

In this section:

1.1.3.1 ToString() Method

The equivalent to DateTime.ToString() with the time zone offset appended to the end.

Syntax

Visual Basic

```
Public Overrides Function ToString () As String
```

C#

```
public override string ToString ()
```

1.1.3.2 ToString(IFormatProvider) Method

The equivalent to DateTime.ToString(provider) with the time zone offset appended to the end.

Syntax

Visual Basic

```
Public Function ToString (ByVal provider As IFormatProvider) As String
```

C#

```
public string ToString (IFormatProvider provider)
```

1.1.3.3 ToString(string) Method

The equivalent to DateTime.ToString(format) with the time zone offset appended to the end.

Syntax

Visual Basic

```
Public Function ToString (ByVal format As String) As String
```

C#

```
public string ToString (string format)
```

1.1.3.4 ToString(string, IFormatProvider) Method

The equivalent to DateTime.ToString(format, provider) with the time zone offset appended to the end.

Syntax

Visual Basic

```
Public Function ToString (
    ByVal format As String,
    ByVal provider As IFormatProvider
) As String
```

C#

```
public string ToString (
    string format,
    IFormatProvider provider
)
```

1.1.4 DateTime Property

Gets the DateTime that corresponds to this without the time zone offset.

Syntax

Microsoft Visual Basic

```
Public Property DateTime As Date
```

C#

```
public DateTime DateTime {get;set;}
```

1.1.5 Day Property

The equivalent to DateTime.Day.

Syntax

Visual Basic

```
Public ReadOnly Property Day As Integer
```

C#

```
public int Day {get;}
```

1.1.6 Hour Property

The equivalent to DateTime.Hour.

Syntax

Visual Basic

```
Public ReadOnly Property Hour As Integer
```

C#

```
public int Hour {get;}
```

1.1.7 Millisecond Property

The equivalent to DateTime.Millisecond.

Syntax

Visual Basic

```
Public ReadOnly Property Millisecond As Integer
```

C#

```
public int Millisecond {get;}
```

1.1.8 Minute Property

The equivalent to DateTime.Minute.

Syntax

Visual Basic

```
Public ReadOnly Property Minute As Integer
```

C#

```
public int Minute {get;}
```

1.1.9 Month Property

The equivalent to DateTime.Month.

Syntax

Visual Basic

```
Public ReadOnly Property Month As Integer
```

C#

```
public int Month {get;}
```

1.1.10 Second Property

The equivalent to DateTime.Second.

Syntax

Visual Basic

```
Public ReadOnly Property Second As Integer
```

C#

```
public int Second {get;}
```

1.1.11 TimeZoneHour Property

Gets the hours part of the time zone offset.

Syntax

Visual Basic

```
Public Property TimeZoneHour As Integer
```

C#

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

1.1.12 TimeZoneMinute Property

Gets the minutes part of the time zone offset.

Syntax

Visual Basic

```
Public Property TimeZoneMinute As Integer
```

C#

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

1.1.13 Year Property

The equivalent to DateTime.Year.

Syntax

Visual Basic

```
Public ReadOnly Property Year As Integer
```

C#

```
public int Year {get;}
```

1.2 DBCommand Interface

Represents a SQL statement or database command.

Syntax

Visual Basic

```
Public Interface DBCommand
```

C#

```
public interface DBCommand
```

Members

All members of DBCommand, including inherited members.

Methods

Modifier and Type	Method	Description
public void	Close() [page 21]	Closes the current SQL statement or command.
public int	ExecuteNonQuery() [page 21]	Executes a non-query statement.
public DBRowReader	ExecuteReader() [page 22]	Executes a query statement returning the result set.
public void	Prepare() [page 22]	Prepares the SQL statement stored in CommandText for execution.

Properties

Modifier and Type	Property	Description
public string	CommandText [page 22]	The SQL statement to be executed.
public DBParameterCollection	Parameters [page 23]	Gets the DBParameterCollection for this DBCommand.

Remarks

DBCommand can represent an update or query.

Example

The following C# code uses the DBCommand interface to execute two queries:

```
DBCommand stmt = conn.CreateCommand();
stmt.CommandText = "SELECT t1a1, t1a2 FROM table1 ";
DBRowReader rs = stmt.ExecuteReader();
printResultSet( rs );
rs.Close();
stmt.CommandText = "SELECT t2a1 FRONT table2 ";
rs = stmt.ExecuteReader();
printResultSet( rs );
rs.Close();
stmt.Close();
```

The following C# code uses the DBCommand interface to execute an update with parameters:

```
public void prepare_for_download(DateTime last_download,
   String ml_username)
{
    DBCommand cstmt = conn.CreateCommand();
    cstmt.CommandText = "CALL myProc( ?,?,? )";
    cstmt.Prepare();
    DBParameter param = new DBParameter();
    param.DbType          = SQLType.SQL_CHAR;
    param.Value           = "10000";
    cstmt.Parameters.Add(param);
    param                 = new DBParameter();
    param.DbType          = SQLType.SQL_INTEGER;
    param.Value           = 20000;
    cstmt.Parameters.Add(param);
    param                 = new DBParameter();
    param.DbType          = SQLType.SQL_DECIMAL;
    param.Precision       = 5;
    param.Value           = new Decimal(30000);
    cstmt.Parameters.Add(param);
    // Execute update
    DBRowReader rset = cstmt.ExecuteNonQuery();
    cstmt.Close();
}
```

In this section:

Closes the current SQL statement or command.

ExecuteNonQuery() Method [page 21]
Executes a non-query statement.

ExecuteReader() Method [page 22]
Executes a query statement returning the result set.

Prepare() Method [page 22]
Prepares the SQL statement stored in CommandText for execution.

CommandText Property [page 22]
The SQL statement to be executed.

Parameters Property [page 23]
Gets the DBParameterCollection for this DBCommand.

1.2.1 Close() Method

Closes the current SQL statement or command.

Syntax

Visual Basic

```
Public Sub Close ()
```

C#

```
public void Close ()
```

1.2.2 ExecuteNonQuery() Method

Executes a non-query statement.

Syntax

Visual Basic

```
Public Function ExecuteNonQuery () As Integer
```

C#

```
public int ExecuteNonQuery ()
```

Returns

The number of rows in the database affected by the SQL statement.

1.2.3 ExecuteReader() Method

Executes a query statement returning the result set.

Syntax

Visual Basic

```
Public Function ExecuteReader () As DBRowReader
```

C#

```
public DBRowReader ExecuteReader ()
```

Returns

A DBRowReader for retrieving results returned by the SQL statement.

1.2.4 Prepare() Method

Prepares the SQL statement stored in CommandText for execution.

Syntax

Visual Basic

```
Public Sub Prepare ()
```

C#

```
public void Prepare ()
```

1.2.5 CommandText Property

The SQL statement to be executed.

Syntax

Visual Basic

```
Public Property CommandText As String
```

C#

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

1.2.6 Parameters Property

Gets the DBParameterCollection for this DBCommand.

Syntax

Visual Basic

```
Public ReadOnly Property Parameters As DBParameterCollection
```

C#

```
public DBParameterCollection Parameters {get;}
```

Returns

The requested Parameter collection.

Related Information

DBParameterCollection Class [page 40]

1.3 DBConnection Interface

Represents a MobiLink ODBC connection.

Syntax

Visual Basic

```
Public Interface DBConnection
```

C#

```
public interface DBConnection
```

Members

All members of DBConnection, including inherited members.

Methods

Modifier and Type	Method	Description
public void	Close() [page 24]	Closes the current connection.
public void	Commit() [page 25]	Commits the current transaction.
public DBCommand	CreateCommand() [page 25]	Creates a SQL statement or command on this connection.
public void	Rollback() [page 25]	Rolls back the current transaction.

Remarks

This interface allows user-written synchronization logic to access an ODBC connection created by MobiLink.

In this section:

Close() Method [page 24]
Closes the current connection.

Commit() Method [page 25]
Commits the current transaction.

CreateCommand() Method [page 25]
Creates a SQL statement or command on this connection.

Rollback() Method [page 25]
Rolls back the current transaction.

1.3.1 Close() Method

Closes the current connection.

Syntax

Visual Basic

```
Public Sub Close ()
```

C#

```
public void Close ()
```

1.3.2 Commit() Method

Commits the current transaction.

Syntax

Visual Basic

```
Public Sub Commit ()
```

C#

```
public void Commit ()
```

1.3.3 CreateCommand() Method

Creates a SQL statement or command on this connection.

Syntax

Visual Basic

```
Public Function CreateCommand () As DBCommand
```

C#

```
public DBCommand CreateCommand ()
```

Returns

The newly generated DBCommand.

1.3.4 Rollback() Method

Rolls back the current transaction.

Syntax

Visual Basic

```
Public Sub Rollback ()
```

C#

```
public void Rollback ()
```

1.4 DBConnectionContext Interface

Obtains information about the current database connection.

Syntax

Visual Basic

```
Public Interface DBConnectionContext
```

C#

```
public interface DBConnectionContext
```

Members

All members of DBConnectionContext, including inherited members.

Methods

Modifier and Type	Method	Description
public DBConnection	GetConnection() [page 28]	Returns the existing connection to the MobiLink consolidated database.
public DownloadData	GetDownloadData() [page 28]	Returns the DownloadData for the current synchronization.
public NameValueCollection	GetProperties() [page 29]	Returns a collection of properties based on this connections script version.
public string	GetRemoteID() [page 30]	Returns the remote ID of the database currently synchronizing on this connection.
public ServerContext	GetServerContext() [page 31]	Returns the current server context.
public string	GetVersion() [page 32]	Returns the version string for this connection.

Properties

Modifier and Type	Property	Description
public NetworkData	NetworkData [page 33]	Returns information about the network streams for a synchronization.

Remarks

This is passed to the constructor of classes containing scripts. If context is required for a background thread or beyond the lifetime of a connection, use a ServerContext.

For more information about constructors, seeConstructors.

> **i** Note
>
> A DBConnectionContext instance should not be used outside the thread that calls into your .NET code.

Example

The following example shows you how to create a class level DBConnectionContext instance to use in your synchronization scripts. The DBConnectionContext getConnection method obtains a DBConnection instance representing the current connection with the MobiLink consolidated database.

```
using Sap.MobiLink.Script;
using System.Data;
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   // The method used for the handle_DownloadData event.
   public void HandleEvent() {
       DBConnection my_connection = _cc.GetConnection();
       // ...
   }
   // ...
}
```

In this section:

1.4.1 GetConnection() Method

Returns the existing connection to the MobiLink consolidated database.

Syntax

Visual Basic

```
Public Function GetConnection () As DBConnection
```

C#

```
public DBConnection GetConnection ()
```

Returns

The current connection to the consolidated database. This connection is only valid for the lifetime of the underlying MobiLink connection.

Remarks

This is the same connection that MobiLink uses to execute SQL scripts. It must not be committed, closed, or altered in any way that would affect the MobiLink server use of the connection.

Do not use the connection after the end_connection event has been called for the connection.

Use the MakeConnection method if a server connection with full access is required.

Related Information

MakeConnection() Method [page 88]

1.4.2 GetDownloadData() Method

Returns the DownloadData for the current synchronization.

Syntax

Visual Basic

```
Public Function GetDownloadData () As DownloadData
```

C#

```
public DownloadData GetDownloadData ()
```

Returns

The DownloadData for the current synchronization; otherwise, null if the synchronization is upload-only.

Remarks

Use the DownloadData instance to create the download for direct row handling.

Example

The following example assumes you have created a DBConnectionContext instance named _cc:

```
// The method used for the handle_DownloadData event.
public void HandleDownload() {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.GetDownloadData();
   // Get an array of tables to set download operations.
        DownloadTableData[] download_tables = my_dd.GetDownloadTables();
        // Get the first table in the DownloadTableData array.
   DownloadTableData my_download_table = download_tables[0];
   // ...
}
```

1.4.3 GetProperties() Method

Returns a collection of properties based on this connections script version.

Syntax

Visual Basic

```
Public Function GetProperties () As NameValueCollection
```

C#

```
public NameValueCollection GetProperties ()
```

Returns

The properties for the current script version.

Remarks

Properties are stored in the ml_property table. For more information, see ml_add_property system procedure.

Example

The following example shows you how to output the properties for a DBConnectionContext. This example assumes you have a DBConnectionContext instance named _cc.

```
// The method used to output the connection properties.
public void OutputProperties() {
   // output the Properties for the current synchronization
   NameValueCollection properties = _cc.GetProperties();
   System.Console.WriteLine(properties.ToString());
}
```

1.4.4 GetRemoteID() Method

Returns the remote ID of the database currently synchronizing on this connection.

Syntax

Visual Basic

```
Public Function GetRemoteID () As String
```

C#

```
public string GetRemoteID ()
```

Returns

The remote ID.

Remarks

For more information about Remote IDs, seeRemote IDs.

Example

The following example shows you how to output the remote ID for a DBConnectionContext:

```
// The method used to output the remote ID.
public void OutputRemoteID() {
   // output the Remote ID for the current synchronization
   string remoteID = _cc.GetRemoteID();
   System.Console.WriteLine(remoteID);
}
```

1.4.5 GetServerContext() Method

Returns the current server context.

Syntax

Visual Basic

```
Public Function GetServerContext () As ServerContext
```

C#

```
public ServerContext GetServerContext ()
```

Returns

The ServerContext for this MobiLink server.

Remarks

This method can be used to create new connections or interact with boot classes.

Example

The following example shows you how to get the ServerContext instance for a DBConnectionContext and shut down the server:

```
// A method that uses an instance of the ServerContext to shut down the server
public void ShutDownServer() {
   ServerContext context = _cc.GetServerContext();
   context.Shutdown();
}
```

1.4.6 GetVersion() Method

Returns the version string for this connection.

Syntax

Visual Basic

```
Public Function GetVersion () As String
```

C#

```
public string GetVersion ()
```

Returns

The script version name.

Remarks

Properties are stored in the ml_property table. For more information, see ml_add_property system procedure.

Example

The following example shows you how to get the script version and use it to make decisions:

```
public void MyEvent() {
   // ...
   string version = _cc.GetVersion();
   switch( version ) {
   case "My Version 1":
       // ...
```

```
        break;
    case "My Version 2":
        // ...
        break;
    }
}
```

1.4.7 NetworkData Property

Returns information about the network streams for a synchronization.

Syntax

Visual Basic

```
Public ReadOnly Property NetworkData As NetworkData
```

C#

```
public NetworkData NetworkData {get;}
```

Returns

Information about the network streams used for the request, or null if the collection has not been enabled.

Remarks

This method is useful when authenticating against another server in the enterprise that uses the client-side certificate and HTTP headers.

To enable a collection of network stream data, add collect_network_data=1 to your -x switches. This option adds additional per-sync memory overhead to store the data.

Related Information

NetworkData Interface [page 73]

1.5 DBParameter Class

Represents a bound ODBC parameter.

Syntax

Visual Basic

```
Public Class DBParameter
```

C#

```
public class DBParameter
```

Members

All members of DBParameter, including inherited members.

Variables

Modifier and Type	Variable	Description
public bool	HasChanged	Returns whether the parameter has been modified since creation.

Properties

Modifier and Type	Property	Description
public SQLType	DbType [page 36]	The SQLType of this parameter.
public ParameterDirection	Direction [page 36]	The Input/Output direction of this parameter.
public bool	IsNullable [page 37]	True if the parameter can be null; otherwise, false.
public string	ParameterName [page 37]	The name of this parameter.
public uint	Precision [page 38]	The Decimal precision of this parameter.
public short	Scale [page 38]	The resolvable digits of this parameter.
public uint	Size [page 39]	The size of this parameter, measured in bytes.
public object	Value [page 39]	The value of this parameter.

Remarks

This class is required to execute commands with parameters. All parameters must be in place before the command is executed.

Example

The following C# code uses the DBCommand interface to execute an update with parameters:

```
using( DBCommand cstmt = conn.CreateCommand() ) {
    DBCommand cstmt = conn.CreateCommand();
    cstmt.CommandText = "call myProc( ?,?,? )";
    cstmt.Prepare();
    DBParameter param = new DBParameter();
    param.DbType          = SQLType.SQL_CHAR;
    param.Value           = "10000";
    cstmt.Parameters.Add( param );
    param                 = new DBParameter();
    param.DbType          = SQLType.SQL_INTEGER;
    param.Value           = 20000;
    cstmt.Parameters.Add( param );
    param                 = new DBParameter();
    param.DbType          = SQLType.SQL_DECIMAL;
    param.Precision       = 5;
    param.Value           = new Decimal( 30000 );
    cstmt.Parameters.Add( param );
    // Execute update
    DBRowReader rset = cstmt.ExecuteNonQuery();
    cstmt.Close();
}
```

In this section:

1.5.1 DbType Property

The SQLType of this parameter.

Syntax

Visual Basic

```
Public Property DbType As SQLType
```

C#

```
public SQLType DbType {get;set;}
```

Remarks

The default value is SQLType.SQL_TYPE_NULL.

1.5.2 Direction Property

The Input/Output direction of this parameter.

Syntax

Visual Basic

```
Public Property Direction As ParameterDirection
```

C#

```
public ParameterDirection Direction {get;set;}
```

Remarks

The default value is ParameterDirection.Input.

1.5.3 IsNullable Property

True if the parameter can be null; otherwise, false.

Syntax

Visual Basic

```
Public Property IsNullable As Boolean
```

C#

```
public bool IsNullable {get;set;}
```

Remarks

The default value is false.

1.5.4 ParameterName Property

The name of this parameter.

Syntax

Visual Basic

```
Public Property ParameterName As String
```

C#

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

Remarks

The default value is null.

1.5.5 Precision Property

The Decimal precision of this parameter.

Syntax

Visual Basic

```
Public Property Precision As UInteger
```

C#

```
public uint Precision {get;set;}
```

Remarks

This property is only used for SQLType.SQL_NUMERIC and SQLType.SQL_DECIMAL parameters.

The default value is 0.

1.5.6 Scale Property

The resolvable digits of this parameter.

Syntax

Visual Basic

```
Public Property Scale As Short
```

C#

```
public short Scale {get;set;}
```

Remarks

This property is only used for SQLType.SQL_NUMERIC and SQLType.SQL_DECIMAL parameters.

The default value is 0.

1.5.7 Size Property

The size of this parameter, measured in bytes.

Syntax

Visual Basic

```
Public Property Size As UInteger
```

C#

```
public uint Size {get;set;}
```

Remarks

The default value is inferred from DbType.

1.5.8 Value Property

The value of this parameter.

Syntax

Visual Basic

```
Public Property Value As Object
```

C#

```
public object Value {get;set;}
```

Remarks

The default value is null.

1.6 DBParameterCollection Class

Collection of DBParameters.

Syntax

Visual Basic

```
Public Class DBParameterCollection Implements
System.Data.IDataParameterCollection, System.Collections.IList,
System.Collections.ICollection, System.Collections.IEnumerable
```

C#

```
public class DBParameterCollection : System.Data.IDataParameterCollection,
System.Collections.IList, System.Collections.ICollection,
System.Collections.IEnumerable
```

Members

All members of DBParameterCollection, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	DBParameterCollection() [page 42]	Creates an empty list of DBParameters.

Methods

Modifier and Type	Method	Description
public int	Add(object) [page 42]	Adds the given parameter to the collection.
public void	Clear() [page 43]	Removes all parameters from the collection.
public bool	Contains [page 43]	Returns true if the collection contains the given DBParameter.
public void	CopyTo(Array, int) [page 45]	Copies the contents of the collection into the given array starting at the specified index.
public IEnumerator	GetEnumerator() [page 46]	Returns an enumerator for the collection.
public int	IndexOf [page 46]	Returns the index of the given DBParameter in the collection.
public void	Insert(int, object) [page 48]	Inserts the given DBParameter into the collection at the specified index.

Modifier and Type	Method	Description
public void	Remove(object) [page 49]	Removes the given DBParameter from the collection.
public void	RemoveAt [page 49]	Removes the DBParameter at the given index in the collection.

Properties

Modifier and Type	Property	Description
public int	Count [page 51]	The number of parameters in the collection.
public bool	IsFixedSize [page 51]	Returns false.
public bool	IsReadOnly [page 51]	Returns false.
public bool	IsSynchronized [page 52]	Returns false.
public object	SyncRoot [page 52]	Used to synchronize access to the DBParameterCollection.
public object	this [page 52]	Gets or sets the DBParameter at the given index in the collection.

Remarks

A DBParamterCollection is initially empty when created by DBCommand, and must be filled with appropriate parameters before the DBCommand executes.

In this section:

DBParameterCollection() Constructor [page 42]
Creates an empty list of DBParameters.

Add(object) Method [page 42]
Adds the given parameter to the collection.

Clear() Method [page 43]
Removes all parameters from the collection.

Contains Method [page 43]
Returns true if the collection contains the given DBParameter.

CopyTo(Array, int) Method [page 45]
Copies the contents of the collection into the given array starting at the specified index.

GetEnumerator() Method [page 46]
Returns an enumerator for the collection.

IndexOf Method [page 46]
Returns the index of the given DBParameter in the collection.

Insert(int, object) Method [page 48]
Inserts the given DBParameter into the collection at the specified index.

Remove(object) Method [page 49]
Removes the given DBParameter from the collection.

RemoveAt Method [page 49]
Removes the DBParameter at the given index in the collection.

Count Property [page 51]
The number of parameters in the collection.

IsFixedSize Property [page 51]
Returns false.

IsReadOnly Property [page 51]
Returns false.

IsSynchronized Property [page 52]
Returns false.

SyncRoot Property [page 52]
Used to synchronize access to the DBParameterCollection.

this Property [page 52]
Gets or sets the DBParameter at the given index in the collection.

1.6.1 DBParameterCollection() Constructor

Creates an empty list of DBParameters.

Syntax

Visual Basic

```
Public Sub DBParameterCollection ()
```

C#

```
public DBParameterCollection ()
```

1.6.2 Add(object) Method

Adds the given parameter to the collection.

Syntax

Visual Basic

```
Public Function Add (ByVal value As Object) As Integer
```

C#

```
public int Add (object value)
```

Parameters

value The DBParameter object to add to the collection.

Returns

The index of the added parameter in the collection.

Related Information

DBParameter Class [page 34]

1.6.3 Clear() Method

Removes all parameters from the collection.

Syntax

Visual Basic

```
Public Sub Clear ()
```

C#

```
public void Clear ()
```

1.6.4 Contains Method

Returns true if the collection contains the given DBParameter.

Overload list

Modifier and Type	Overload name	Description
public bool	Contains(object) [page 44]	Returns true if the collection contains the given DBParameter.

Modifier and Type	Overload name	Description
public bool	Contains(string) [page 45]	Checks whether the collection contains a parameter with the specified name.

In this section:

Contains(object) Method [page 44]
Returns true if the collection contains the given DBParameter.

Contains(string) Method [page 45]
Checks whether the collection contains a parameter with the specified name.

1.6.4.1 Contains(object) Method

Returns true if the collection contains the given DBParameter.

Syntax

Visual Basic

```
Public Function Contains (ByVal value As Object) As Boolean
```

C#

```
public bool Contains (object value)
```

Parameters

value The DBParameter object to check for.

Returns

True if this collection contains the DBParameter; otherwise, false.

Related Information

DBParameter Class [page 34]

1.6.4.2 Contains(string) Method

Checks whether the collection contains a parameter with the specified name.

Syntax

Visual Basic

```
Public Function Contains (ByVal parameterName As String) As Boolean
```

C#

```
public bool Contains (string parameterName)
```

Parameters

parameterName The name of the parameter to check for.

Returns

True if this collection contains a parameter with the given name; otherwise, false.

1.6.5 CopyTo(Array, int) Method

Copies the contents of the collection into the given array starting at the specified index.

Syntax

Visual Basic

```
Public Sub CopyTo (
    ByVal array As Array,
    ByVal index As Integer
)
```

C#

```
public void CopyTo (
    Array array,
    int index
)
```

Parameters

array The array to which the contents of the collection are copied into.
index The index in the array at which the contents of the array should be copied into the collection.

1.6.6 GetEnumerator() Method

Returns an enumerator for the collection.

Syntax

Visual Basic

```
Public Function GetEnumerator () As System.Collections.IEnumerator
```

C#

```
public IEnumerator GetEnumerator ()
```

Returns

An enumerator for the collection.

1.6.7 IndexOf Method

Returns the index of the given DBParameter in the collection.

Overload list

Modifier and Type	Overload name	Description
public int	IndexOf(object) [page 47]	Returns the index of the given DBParameter in the collection.
public int	IndexOf(string) [page 47]	Returns the index of the parameter with the given name in the collection.

In this section:

IndexOf(object) Method [page 47]
Returns the index of the given DBParameter in the collection.

IndexOf(string) Method [page 47]
Returns the index of the parameter with the given name in the collection.

1.6.7.1 IndexOf(object) Method

Returns the index of the given DBParameter in the collection.

Syntax

Visual Basic

```
Public Function IndexOf (ByVal value As Object) As Integer
```

C#

```
public int IndexOf (object value)
```

Parameters

value The DBParameter object to find.

Returns

The index of the DBParameter in the collection.

Related Information

DBParameter Class [page 34]

1.6.7.2 IndexOf(string) Method

Returns the index of the parameter with the given name in the collection.

Syntax

Visual Basic

```
Public Function IndexOf (ByVal parameterName As String) As Integer
```

C#

```
public int IndexOf (string parameterName)
```

Parameters

parameterName The name of the parameter to find.

Returns

The index of the parameter, or -1 if there is no parameter with the given name.

1.6.8 Insert(int, object) Method

Inserts the given DBParameter into the collection at the specified index.

Syntax

Visual Basic

```
Public Sub Insert (
    ByVal index As Integer,
    ByVal value As Object
)
```

C#

```
public void Insert (
    int index,
    object value
)
```

Parameters

value The DBParameter object to insert.
index The index at which to insert the value.

Related Information

DBParameter Class [page 34]

1.6.9 Remove(object) Method

Removes the given DBParameter from the collection.

Syntax

Visual Basic

```
Public Sub Remove (ByVal value As Object)
```

C#

```
public void Remove (object value)
```

Parameters

value The DBParameter to remove.

Related Information

DBParameter Class [page 34]

1.6.10 RemoveAt Method

Removes the DBParameter at the given index in the collection.

Overload list

Modifier and Type	Overload name	Description
public void	RemoveAt(int) [page 50]	Removes the DBParameter at the given index in the collection.
public void	RemoveAt(string) [page 50]	Removes the parameter with the given name from the collection.

In this section:

RemoveAt(int) Method [page 50]
Removes the DBParameter at the given index in the collection.

RemoveAt(string) Method [page 50]

Removes the parameter with the given name from the collection.

1.6.10.1 RemoveAt(int) Method

Removes the DBParameter at the given index in the collection.

Syntax

Visual Basic

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

C#

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

Parameters

index The index of the DBParameter to remove.

Related Information

DBParameter Class [page 34]

1.6.10.2 RemoveAt(string) Method

Removes the parameter with the given name from the collection.

Syntax

Visual Basic

```
Public Sub RemoveAt (ByVal parameterName As String)
```

C#

```
public void RemoveAt (string parameterName)
```

Parameters

parameterName The name of the parameter to remove.

1.6.11 Count Property

The number of parameters in the collection.

Syntax

Visual Basic

```
Public ReadOnly Property Count As Integer
```

C#

```
public int Count {get;}
```

1.6.12 IsFixedSize Property

Returns false.

Syntax

Visual Basic

```
Public ReadOnly Property IsFixedSize As Boolean
```

C#

```
public bool IsFixedSize {get;}
```

1.6.13 IsReadOnly Property

Returns false.

Syntax

Visual Basic

```
Public ReadOnly Property IsReadOnly As Boolean
```

C#

```
public bool IsReadOnly {get;}
```

1.6.14 IsSynchronized Property

Returns false.

Syntax

Visual Basic

```
Public ReadOnly Property IsSynchronized As Boolean
```

C#

```
public bool IsSynchronized {get;}
```

1.6.15 SyncRoot Property

Used to synchronize access to the DBParameterCollection.

Syntax

Visual Basic

```
Public ReadOnly Property SyncRoot As Object
```

C#

```
public object SyncRoot {get;}
```

1.6.16 this Property

Gets or sets the DBParameter at the given index in the collection.

Overload list

Modifier and Type	Overload name	Description
public object	this[int index] [page 53]	Gets or sets the DBParameter at the given index in the collection.
public object	this[string parameterName] [page 53]	Gets or sets the DBParameter with the given name in the collection.

In this section:

Gets or sets the DBParameter at the given index in the collection.

this[string parameterName] Property [page 53]
Gets or sets the DBParameter with the given name in the collection.

1.6.16.1 this[int index] Property

Gets or sets the DBParameter at the given index in the collection.

Syntax

Visual Basic

```
Public Property Item (ByVal index As Integer) As Object
```

C#

```
public object this[int index] {get;set;}
```

Returns

This with the given index in the collection.

Related Information

DBParameter Class [page 34]

1.6.16.2 this[string parameterName] Property

Gets or sets the DBParameter with the given name in the collection.

Syntax

Visual Basic

```
Public Property Item (ByVal parameterName As String) As Object
```

C#

```
public object this[string parameterName] {get;set;}
```

Returns

This with the given name in the collection.

Related Information

DBParameter Class [page 34]

1.7 DBRowReader Interface

Represents a set of rows being read from a database.

Syntax

Visual Basic

```
Public Interface DBRowReader
```

C#

```
public interface DBRowReader
```

Members

All members of DBRowReader, including inherited members.

Methods

Modifier and Type	Method	Description
public void	Close() [page 55]	Cleans up all resources used by this MLDBRowReader.
public object[]	NextRow() [page 56]	Retrieves and returns the next row in the result set.

Properties

Modifier and Type	Property	Description
public string[]	ColumnNames [page 56]	Gets the names of all columns in the result set.
public SQLType[]	ColumnTypes [page 57]	Gets the types of all columns in the result set.

Remarks

Executing the ExecuteReader method creates a DBRowReader.

Example

The following C# code calls a function with the rows in the result set represented by the given DBRowReader:

```
DBCommand stmt = conn.CreateCommand();
stmt.CommandText = "select intCol, strCol from table1 ";
DBRowReader rs = stmt.ExecuteReader();
object[] values = rset.NextRow();
while( values != null ) {
    handleRow( (int)values[0], (String)values[1] );
    values = rset.NextRow();
}
rset.Close();
stmt.Close();
```

In this section:

Related Information

ExecuteReader() Method [page 22]

1.7.1 Close() Method

Cleans up all resources used by this MLDBRowReader.

Syntax

Visual Basic

```
Public Sub Close ()
```

C#

```
public void Close ()
```

Remarks

This MLDBRowReader cannot be used again after this method is called.

1.7.2 NextRow() Method

Retrieves and returns the next row in the result set.

Syntax

Visual Basic

```
Public Function NextRow () As Object()
```

C#

```
public object[] NextRow ()
```

Returns

The next row of values in the result set, or null if there are no more rows in this result set.

Related Information

SQLType Enumeration [page 113]

1.7.3 ColumnNames Property

Gets the names of all columns in the result set.

Syntax

Visual Basic

```
Public ReadOnly Property ColumnNames As String()
```

C#

```
public string[] ColumnNames {get;}
```

Remarks

The value is an array of strings corresponding to the column names in the result set.

1.7.4 ColumnTypes Property

Gets the types of all columns in the result set.

Syntax

Visual Basic

```
Public ReadOnly Property ColumnTypes As SQLType()
```

C#

```
public SQLType[] ColumnTypes {get;}
```

Remarks

The value is an array of SQLTypes corresponding to the column types in the result set.

1.8 DownloadData Interface

Encapsulates all download data operations for direct row handling.

Syntax

Visual Basic

```
Public Interface DownloadData
```

C#

```
public interface DownloadData
```

Members

All members of DownloadData, including inherited members.

Methods

Modifier and Type	Method	Description
public DownloadTableData	GetDownloadTableByName(string) [page 59]	Gets the named download table for this synchronization.
public DownloadTableData[]	GetDownloadTables() [page 59]	Gets an array of all the tables for download in this synchronization.

Remarks

Use the GetDownloadData method To obtain a DownloadData instance. Use the GetDownloadTables and GetDownloadTableByName methods to return DownloadTableData instances.

This download data is available through DBConnectionContext. It is not valid to access the download data before the begin_sync event, or the DownloadData in an upload only synchronization.

For more information about direct row handling, seehandle_DownloadData connection event andDirect row handling.

In this section:

GetDownloadTableByName(string) Method [page 59]
Gets the named download table for this synchronization.

GetDownloadTables() Method [page 59]
Gets an array of all the tables for download in this synchronization.

Related Information

DownloadTableData Interface [page 60]
GetDownloadData() Method [page 28]

1.8.1 GetDownloadTableByName(string) Method

Gets the named download table for this synchronization.

Syntax

Visual Basic

```
Public Function GetDownloadTableByName (ByVal table_name As String) As
DownloadTableData
```

C#

```
public DownloadTableData GetDownloadTableByName (string table_name)
```

Parameters

table_name The name of the table for which you want the download data

Returns

The download data for the given table name, or null if not found.

1.8.2 GetDownloadTables() Method

Gets an array of all the tables for download in this synchronization.

Syntax

Visual Basic

```
Public Function GetDownloadTables () As DownloadTableData()
```

C#

```
public DownloadTableData[] GetDownloadTables ()
```

Returns

An array of download table data. The order of tables in the array is the same as the upload order for the remote.

Remarks

The operations performed on this table are sent to the remote database.

Example

The following example uses the GetDownloadTables method to obtain an array of DownloadTableData objects for the current synchronization. The example assumes you have a DBConnectionContext instance named _cc.

```
// The method used for the handle_DownloadData event.
public void HandleDownload() {
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.GetDownloadData();
   // Get an array of tables to set download operations.
   DownloadTableData[] download_tables = my_dd.GetDownloadTables();
   // Get the first table in the DownloadTableData array.
   DownloadTableData my_download_table = download_tables[0];
   // ...
}
```

1.9 DownloadTableData Interface

Encapsulates information for one download table for a synchronization.

Syntax

Visual Basic

```
Public Interface DownloadTableData
```

C#

```
public interface DownloadTableData
```

Members

All members of DownloadTableData, including inherited members.

Methods

Modifier and Type	Method	Description
public IDbCommand	GetDeleteCommand() [page 62]	Get a command which allows the user to add delete operations to the download data operations.

Modifier and Type	Method	Description
public DateTime	GetLastDownloadTime() [page 63]	Returns the last download time for this table.
public string	GetName() [page 64]	Gets the table name of this instance.
public DataTable	GetSchemaTable() [page 64]	Gets a DataTable that describes the metadata for this download table.
public IDbCommand	GetUpsertCommand() [page 65]	Get a command which allows the user to add upsert(insert/update) operations to the download data operations.

Remarks

Use this interface to set the data operations that are downloaded to a synchronization client site.

Example

Suppose you have the following table:

```
CREATE TABLE remoteOrders (
   pk INT NOT NULL,
   col1 VARCHAR(200),
   PRIMARY KEY (pk)
);
```

The following example uses the GetDownloadTableByName method to return a DownloadTableData instance representing the remoteOrders table:

```
// The method used for the handle_DownloadData event
public void HandleDownload() {
   // _cc is a DBConnectionContext instance.
   // Get the DownloadData for the current synchronization.
   DownloadData my_dd = _cc.GetDownloadData();
   // Get the DownloadTableData for the remoteOrders table.
   DownloadTableData td = my_dd.GetDownloadTableByName("remoteOrders");
   // User defined-methods to set download operations.
   SetDownloadUpserts(td);
   SetDownloadDeletes(td);
   // ...
}
```

In this example, the SetDownloadInserts method uses GetUpsertCommand to obtain a command for the rows you want to insert or update. The IDbCommand holds the parameters that you set to the values you want inserted on the remote database.

```
void SetDownloadInserts(DownloadTableData td) {
   IDbCommand  upsert_cmd = td.GetUpsertCommand();
   IDataParameterCollection parameters = upsert_cmd.Parameters;
   // The following method calls are the same as the following SQL statement:
   // INSERT INTO remoteOrders(pk, col1) values(2300, "truck");
   ((IDataParameter) (parameters[0])).Value = (Int32) 2300;
```

```
    ((IDataParameter) (parameters[1])).Value = (String) "truck";
    if (upsert_cmd.ExecuteNonQuery() > 0) {
        // Insert was not filtered.
    }
    else {
        // Insert was filtered because it was uploaded
        // in the same synchronization.
    }
}
```

The following method uses the DownloadTableData.GetDeleteCommand to obtain a command for rows you want to delete.

```
void SetDownloadDeletes(DownloadTableData td) {
    IDbCommand  delete_cmd = t2_download_dd.GetDeleteCommand();
    // The following method calls are the same as the following SQL statement:
    // DELETE FROM remoteOrders  where pk = 2300;
    IDataParameterCollection parameters = delete_cmd.Parameters;
    ((IDataParameter) (parameters[0])).Value = (Int32) 2300;
    delete_cmd.ExecuteNonQuery();
}
```

In this section:

1.9.1 GetDeleteCommand() Method

Get a command which allows the user to add delete operations to the download data operations.

Syntax

Visual Basic

```
Public Function GetDeleteCommand () As IDbCommand
```

C#

```
public IDbCommand GetDeleteCommand ()
```

Returns

A command for deletes in the download.

Remarks

The command returned has the same number of parameters as primary key columns in this table. The column values for the primary key columns must be set and the statement executed with ExecuteNonQuery for the delete to be included in the download. ExecuteNonQuery on the command returns 0 if the delete operation was filtered and returns 1 if the insert was not filtered.

To delete a row, you must set all primary key values for download delete operations. To truncate the remote table, set all primary key columns to null.

Related Information

DownloadTableData Interface [page 60]

1.9.2 GetLastDownloadTime() Method

Returns the last download time for this table.

Syntax

Visual Basic

```
Public Function GetLastDownloadTime () As Date
```

C#

```
public DateTime GetLastDownloadTime ()
```

Returns

The last download time for this table.

Remarks

This is the same last download time passed to several of the per table download events.

The last download time is useful for generating the table download data for a particular synchronization.

1.9.3 GetName() Method

Gets the table name of this instance.

Syntax

Visual Basic

```
Public Function GetName () As String
```

C#

```
public string GetName ()
```

Returns

The table name of this instance.

Remarks

This is a utility function. The table name can also be accessed via the Schema for this instance.

1.9.4 GetSchemaTable() Method

Gets a DataTable that describes the metadata for this download table.

Syntax

Visual Basic

```
Public Function GetSchemaTable () As DataTable
```

C#

```
public DataTable GetSchemaTable ()
```

Returns

A DataTable that describes the column metadata.

Remarks

You must specify the client option to send column names if you want the DataTable to contain column name information. Send column names is specified by default.

1.9.5 GetUpsertCommand() Method

Get a command which allows the user to add upsert (insert/update) operations to the download data operations.

Syntax

Visual Basic

```
Public Function GetUpsertCommand () As IDbCommand
```

C#

```
public IDbCommand GetUpsertCommand ()
```

Returns

A command for inserts/updates for the download.

Remarks

The command returned has the same number of parameters as columns in this table. The column values for the insert must be set and the statement executed with ExecuteNonQuery for the insert/update to be included in the download. ExecuteNonQuery on the command returns 0 if the insert operation was filtered and returns 1 if the insert was not filtered.

You cannot add or remove parameters to this command; you can only set their values.

Related Information

DownloadTableData Interface [page 60]

1.10 FatalException Class

Signals MobiLink that a fatal server-side error has occurred internally and should shutdown immediately.

Syntax

Visual Basic

```
Public Class FatalException Inherits System.ApplicationException
```

C#

```
public class FatalException : System.ApplicationException
```

Members

All members of FatalException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	FatalException [page 67]	Creates a new FatalException with the default message.

In this section:

FatalException Constructor [page 67]
Creates a new FatalException with the default message.

1.10.1 FatalException Constructor

Creates a new FatalException with the default message.

Overload list

Modifier and Type	Overload name	Description
public	FatalException() [page 67]	Creates a new FatalException with the default message.
protected	FatalException(SerializationInfo, StreamingContext) [page 68]	Creates a new FatalException from the given SerializationInfo and StreamingContext objects.
public	FatalException(string) [page 68]	Creates a new FatalException with the given message.
public	FatalException(string, Exception) [page 69]	Creates a new FatalException with the given message and containing the given inner exception that caused this one.

In this section:

FatalException() Constructor [page 67]
Creates a new FatalException with the default message.

FatalException(SerializationInfo, StreamingContext) Constructor [page 68]
Creates a new FatalException from the given SerializationInfo and StreamingContext objects.

FatalException(string) Constructor [page 68]
Creates a new FatalException with the given message.

FatalException(string, Exception) Constructor [page 69]
Creates a new FatalException with the given message and containing the given inner exception that caused this one.

1.10.1.1 FatalException() Constructor

Creates a new FatalException with the default message.

Syntax

Visual Basic

```
Public Sub FatalException ()
```

C#

```
public FatalException ()
```

1.10.1.2 FatalException(SerializationInfo, StreamingContext) Constructor

Creates a new FatalException from the given SerializationInfo and StreamingContext objects.

Syntax

Visual Basic

```
Protected Sub FatalException (
    ByVal info As SerializationInfo,
    ByVal context As StreamingContext
)
```

C#

```
protected FatalException (
    SerializationInfo info,
    StreamingContext context
)
```

Parameters

info The SerializationInfo object to use to construct this FatalException.
context The StreamingContext object to use to construct this FatalException.

1.10.1.3 FatalException(string) Constructor

Creates a new FatalException with the given message.

Syntax

Visual Basic

```
Public Sub FatalException (ByVal message As String)
```

C#

```
public FatalException (string message)
```

Parameters

message The message for this FatalException.

1.10.1.4 FatalException(string, Exception) Constructor

Creates a new FatalException with the given message and containing the given inner exception that caused this one.

Syntax

Visual Basic

```
Public Sub FatalException (
    ByVal message As String,
    ByVal ie As Exception
)
```

C#

```
public FatalException (
    string message,
    Exception ie
)
```

Parameters

message The message for this FatalException.
ie The exception that caused this FatalException.

1.11 LogMessage Class

Contains information regarding a message printed to the log.

Syntax

Microsoft Visual Basic

```
Public Class LogMessage
```

C#

```
public class LogMessage
```

Members

All members of LogMessage, including inherited members.

Enumerations

Modifier and Type	Enumeration	Description
Public	MessageType [page 71]	Enumerates all possible LogMessage types.

Constructors

Modifier and Type	Constructor	Description
public	LogMessage(MessageType, string, string) [page 71]	Create a LogMessage with the given attributes.

Properties

Modifier and Type	Property	Description
public string	Text [page 72]	The main text of the message.
public MessageType	Type [page 72]	The type of log message that this instance represents.
public string	User [page 72]	The user for which this message is being logged.

Remarks

An instance of this class is passed into a LogCallback.

In this section:

LogMessage(MessageType, string, string) Constructor [page 71]
Create a LogMessage with the given attributes.

MessageType Enumeration [page 71]
Enumerates all possible LogMessage types.

Text Property [page 72]
The main text of the message.

Type Property [page 72]
The type of log message that this instance represents.

User Property [page 72]
The user for which this message is being logged.

Related Information

LogCallback(ServerContext, LogMessage) Delegate [page 112]

1.11.1 LogMessage(MessageType, string, string) Constructor

Create a LogMessage with the given attributes.

Syntax

Visual Basic

```
Public Sub LogMessage (
    ByVal type As MessageType,
    ByVal user As String,
    ByVal text As String
)
```

C#

```
public LogMessage (
    MessageType type,
    string user,
    string text
)
```

1.11.2 MessageType Enumeration

Enumerates all possible LogMessage types.

Syntax

Visual Basic

```
Public Enum MessageType
```

C#

```
enum MessageType
```

Members

Member name	Description
ERROR	A log message is an error.
WARNING	A log message is a warning.
INFO	A log information message.

1.11.3 Text Property

The main text of the message.

Syntax

Visual Basic

```
Public ReadOnly Property Text As String
```

C#

```
public string Text {get;}
```

1.11.4 Type Property

The type of log message that this instance represents.

Syntax

Visual Basic

```
Public ReadOnly Property Type As MessageType
```

C#

```
public MessageType Type {get;}
```

1.11.5 User Property

The user for which this message is being logged.

Syntax

Visual Basic

```
Public ReadOnly Property User As String
```

C#

```
public string User {get;}
```

Remarks

This property can be null.

1.12 NetworkData Interface

Contains information about the network streams for a synchronization.

Syntax

Visual Basic

```
Public Interface NetworkData
```

C#

```
public interface NetworkData
```

Members

All members of NetworkData, including inherited members.

Methods

Modifier and Type	Method	Description
public string	GetHTTPHeaderValue(string) [page 75]	Returns the value of the last header received by the server with the supplied name.
public IList< string >	GetHTTPHeaderValues(string) [page 76]	Returns all the header values received by the server associated with the supplied name.

Properties

Modifier and Type	Property	Description
public X509Certificate2Collection	ClientCertificates [page 77]	Returns an X509Certificate2Collection containing any certificates sent by the client.
public IDictionary< string, IList< string > >	HTTPHeaders [page 77]	Returns a dictionary that maps header names to a list of header values.
public bool	IsEndToEndEncrypted [page 78]	Determines if this synchronization is end-to-end encrypted.
public bool	IsHTTP [page 78]	Determines if the synchronization uses HTTP or HTTPS.
public bool	IsTLS [page 79]	Determines if this synchronization uses TLS.

Remarks

This interface is useful when authenticating against another server in the enterprise that uses the client-side certificate and HTTP headers.

To enable a collection of network stream data, add collect_network_data=1 to your -x switches. This option adds additional per-sync memory overhead to store the data. When using TLS or HTTPS with client-side certificates, add trusted_certificates=<certificate file> to have the server ask the client to send a certificate during the TLS handshake, incurring a time and network cost.

You can obtain a NetworkData object by invoking the NetworkData method of the DBConnectionContext interface. When using HTTP or HTTPS, it contains the header data for the last HTTP request received by the server before the authenticate scripts are invoked.

Example

The following example illustrates how to get a NetworkData object from the DBConnectionContext object, and output the data.

```
using Sap.MobiLink.Script;
using System.Collections.Generic;
using System.Security.Cryptography.X509Certificates;
public class OrderProcessor {
   DBConnectionContext _cc;
   public OrderProcessor( DBConnectionContext cc ) {
       _cc = cc;
   }
   public void AuthUser() {
       NetworkData nd = _cc.NetworkData;
       if( nd != null ) {
           if( nd.IsHTTP ) {
               PrintLn( "http" );
               string user_agent = nd.GetHTTPHeaderValue( "user-agent" );
               PrintLn( "  user-agent: " + user_agent.Substring( 0,
user_agent.IndexOf( '/' ) ) );
           } else {
               PrintLn( "no http" );
           }
           if( nd.IsTLS ) {
               PrintLn( "tls" );
               X509Certificate2Collection certs = nd.ClientCertificates;
               if( certs != null ) {
                   PrintLn( "  client-side cert:" );
                   int n = 1;
                   foreach( X509Certificate2 x509 in certs ) {
                       PrintLn( "    cert " + n++ );
                       PrintLn( "      Subject: " + x509.SubjectName.Name );
                       PrintLn( "      Issuer: " + x509.IssuerName.Name );
                   }
               } else {
                   PrintLn( "  no client cert" );
               }
           } else {
               PrintLn( "no tls" );
           }
           if( nd.IsEndToEndEncrypted ) {
               PrintLn( "e2ee" );
           } else {
               PrintLn( "no e2ee" );
```

```
            }
        } else {
            PrintLn( "NULL networkdata" );
        }
    }
}
```

In this section:

1.12.1 GetHTTPHeaderValue(string) Method

Returns the value of the last header received by the server with the supplied name.

Syntax

Visual Basic

```
Public Function GetHTTPHeaderValue (ByVal name As String) As String
```

C#

```
public string GetHTTPHeaderValue (string name)
```

Parameters

name The header name to return the value for.

Returns

The last header value associated with the supplied header name.

Related Information

GetHTTPHeaderValues(string) Method [page 76]
HTTPHeaders Property [page 77]

1.12.2 GetHTTPHeaderValues(string) Method

Returns all the header values received by the server associated with the supplied name.

Syntax

Visual Basic

```
Public Function GetHTTPHeaderValues (ByVal name As String) As IList<
string >
```

C#

```
public IList< string > GetHTTPHeaderValues (string name)
```

Parameters

name The header name to return the values for.

Returns

The header values associated with the supplied header name.

Related Information

GetHTTPHeaderValue(string) Method [page 75]
HTTPHeaders Property [page 77]

1.12.3 ClientCertificates Property

Returns an X509Certificate2Collection containing any certificates sent by the client.

Syntax

Visual Basic

```
Public ReadOnly Property ClientCertificates As X509Certificate2Collection
```

C#

```
public X509Certificate2Collection ClientCertificates {get;}
```

Returns

An X509Certificate2Collection containing the X509 certificates that identify the client, or null if no such certificates were provided.

Remarks

This function will return a non-null value only if isTLS() is true, and the client supplies a certificate using the "identity" stream parameter, and the trusted_certificates option is set on the server. A non-null CertPath will contain the certificates in order, from the self-signed certificate to the peer certificate.

1.12.4 HTTPHeaders Property

Returns a dictionary that maps header names to a list of header values.

Syntax

Visual Basic

```
Public ReadOnly Property HTTPHeaders As IDictionary< string, IList< string
> >
```

C#

```
public IDictionary< string, IList< string > > HTTPHeaders {get;}
```

Returns

A dictionary of header name-value pairs.

Related Information

GetHTTPHeaderValue(string) Method [page 75]
GetHTTPHeaderValues(string) Method [page 76]

1.12.5 IsEndToEndEncrypted Property

Determines if this synchronization is end-to-end encrypted.

Syntax

Visual Basic

```
Public ReadOnly Property IsEndToEndEncrypted As Boolean
```

C#

```
public bool IsEndToEndEncrypted {get;}
```

1.12.6 IsHTTP Property

Determines if the synchronization uses HTTP or HTTPS.

Syntax

Visual Basic

```
Public ReadOnly Property IsHTTP As Boolean
```

C#

```
public bool IsHTTP {get;}
```

Returns

True if this synchronization uses HTTP or HTTPS; otherwise, returns false.

1.12.7 IsTLS Property

Determines if this synchronization uses TLS.

Syntax

Visual Basic

```
Public ReadOnly Property IsTLS As Boolean
```

C#

```
public bool IsTLS {get;}
```

Returns

True if this synchronization uses TLS; otherwise, returns false.

1.13 ScriptExecutionException Class

Signals that an error has occurred in a user script.

Syntax

Visual Basic

```
Public Class ScriptExecutionException Inherits System.ApplicationException
```

C#

```
public class ScriptExecutionException : System.ApplicationException
```

Members

All members of ScriptExecutionException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ScriptExecutionException [page 80]	Creates a new ScriptExecutionException with the default message.

Remarks

Throwing this exception or any derivations of it, except for SychronizationException, causes the MobiLink server to shut down.

In this section:

ScriptExecutionException Constructor [page 80]
Creates a new ScriptExecutionException with the default message.

1.13.1 ScriptExecutionException Constructor

Creates a new ScriptExecutionException with the default message.

Overload list

Modifier and Type	Overload name	Description
public	ScriptExecutionException() [page 81]	Creates a new ScriptExecutionException with the default message.
protected	ScriptExecutionException(SerializationInfo, StreamingContext) [page 81]	Creates a new ScriptExecutionException from the given SerializationInfo and StreamingContext objects.
public	ScriptExecutionException(string) [page 82]	Creates a new ScriptExecutionException with the given message.
public	ScriptExecutionException(string, Exception) [page 82]	Creates a new ScriptExecutionException with the given message and containing the given inner exception that caused this one.

In this section:

ScriptExecutionException() Constructor [page 81]
Creates a new ScriptExecutionException with the default message.

ScriptExecutionException(SerializationInfo, StreamingContext) Constructor [page 81]
Creates a new ScriptExecutionException from the given SerializationInfo and StreamingContext objects.

ScriptExecutionException(string) Constructor [page 82]
Creates a new ScriptExecutionException with the given message.

ScriptExecutionException(string, Exception) Constructor [page 82]
Creates a new ScriptExecutionException with the given message and containing the given inner exception that caused this one.

1.13.1.1 ScriptExecutionException() Constructor

Creates a new ScriptExecutionException with the default message.

Syntax

Visual Basic

```
Public Sub ScriptExecutionException ()
```

C#

```
public ScriptExecutionException ()
```

1.13.1.2 ScriptExecutionException(SerializationInfo, StreamingContext) Constructor

Creates a new ScriptExecutionException from the given SerializationInfo and StreamingContext objects.

Syntax

Visual Basic

```
Protected Sub ScriptExecutionException (
    ByVal info As SerializationInfo,
    ByVal context As StreamingContext
)
```

C#

```
protected ScriptExecutionException (
    SerializationInfo info,
    StreamingContext context
)
```

Parameters

info The SerializationInfo object to construct this ScriptExecutionException.
context The StreamingContext object to construct this ScriptExecutionException.

1.13.1.3 ScriptExecutionException(string) Constructor

Creates a new ScriptExecutionException with the given message.

Syntax

Visual Basic

```
Public Sub ScriptExecutionException (ByVal message As String)
```

C#

```
public ScriptExecutionException (string message)
```

Parameters

message The message for this ScriptExecutionException.

1.13.1.4 ScriptExecutionException(string, Exception) Constructor

Creates a new ScriptExecutionException with the given message and containing the given inner exception that caused this one.

Syntax

Visual Basic

```
Public Sub ScriptExecutionException (
    ByVal message As String,
    ByVal ie As Exception
)
```

C#

```
public ScriptExecutionException (
    string message,
    Exception ie
)
```

Parameters

message The message for this ScriptExecutionException.
ie The exception that caused this ScriptExecutionException.

1.14 ServerContext Interface

Instantiates the context that is present for the duration of the MobiLink server.

Syntax

Visual Basic

```
Public Interface ServerContext
```

C#

```
public interface ServerContext
```

Members

All members of ServerContext, including inherited members.

Methods

Modifier and Type	Method	Description
public NameValueCollection	getProperties(string, string) [page 85]	Returns a collection of the properties for the given component and set.
public NameValueCollection	getPropertiesByVersion(string) [page 85]	Returns a collection of the properties for the given script version.
public StringCollection	getPropertySetNames(string) [page 86]	Returns a collection of the set names for the given component.
public object[]	GetStartClassInstances() [page 87]	Returns all start classes loaded into this domain.
public DBConnection	MakeConnection() [page 88]	Creates a new database connection.
public void	Shutdown() [page 88]	Causes the MobiLink server to perform a soft shutdown.

Events

Modifier and Type	Event	Description
public LogCallback	ErrorListener [page 89]	Triggered when the MobiLink server prints an error.
public LogCallback	InfoListener [page 89]	Triggered when the MobiLink server prints information.
public ShutdownCallback	ShutdownListener [page 89]	Triggered when the MobiLink server is shutting down.
public LogCallback	WarningListener [page 90]	Triggered when the MobiLink server prints a warning.

Remarks

This context can be held as static data and used in a background thread. It is valid for the duration of the .NET CLR invoked by MobiLink.

Use the GetServerContext method to access a ServerContext instance. It is passed to the constructor of boot classes.

In this section:

Related Information

GetServerContext() Method [page 31]

1.14.1 getProperties(string, string) Method

Returns a collection of the properties for the given component and set.

Syntax

Visual Basic

```
Public Function getProperties (
    ByVal component As String,
    ByVal set As String
) As NameValueCollection
```

C#

```
public NameValueCollection getProperties (
    string component,
    string set
)
```

Parameters

component Refers to the component_name column of the ml_property table.
set Refers to the property_set_name column of the ml_property table.

Returns

The properties for the given component/set.

Remarks

Properties are stored in the ml_property table. For more information, see ml_add_property system procedure.

1.14.2 getPropertiesByVersion(string) Method

Returns a collection of the properties for the given script version.

Syntax

Visual Basic

```
Public Function getPropertiesByVersion (ByVal script_version As String) As
NameValueCollection
```

C#

```
public NameValueCollection getPropertiesByVersion (string script_version)
```

Parameters

script_version The script version for which to return associated properties.

Returns

The properties for the given script version.

Remarks

Properties are stored in the ml_property table. For more information, see ml_add_property system procedure.

1.14.3 getPropertySetNames(string) Method

Returns a collection of the set names for the given component.

Syntax

Visual Basic

```
Public Function getPropertySetNames (ByVal component As String) As
StringCollection
```

C#

```
public StringCollection getPropertySetNames (string component)
```

Parameters

component Refers to the component_name column of the ml_property table.

Returns

The collection of set names for the given component.

Remarks

Properties are stored in the ml_property table. For more information, see ml_add_property system procedure.

1.14.4 GetStartClassInstances() Method

Returns all start classes loaded into this domain.

Syntax

Visual Basic

```
Public Function GetStartClassInstances () As Object()
```

C#

```
public object[] GetStartClassInstances ()
```

Returns

An array of all start classes that were constructed at the server start time. The array length is zero if there are no start classes.

Remarks

For more information about user-defined start classes, see User-defined start classes.

Example

The following example demonstrates how to find a start class:

```
void FindStartClass( ServerContext sc, string name )
{
   object[] startClasses = sc.GetStartClassInstances();
   foreach( object obj in startClasses ) {
```

```
        if( obj is MyClass ) {
            // Execute some code.....
        }
    }
}
```

1.14.5 MakeConnection() Method

Creates a new database connection.

Syntax

Visual Basic

```
Public Function MakeConnection () As DBConnection
```

C#

```
public DBConnection MakeConnection ()
```

Returns

A new connection.

1.14.6 Shutdown() Method

Causes the MobiLink server to perform a soft shutdown.

Syntax

Visual Basic

```
Public Sub Shutdown ()
```

C#

```
public void Shutdown ()
```

1.14.7 ErrorListener Event

Triggered when the MobiLink server prints an error.

Syntax

Visual Basic

```
Public Event ErrorListener  As LogCallback
```

C#

```
public LogCallback ErrorListener;
```

1.14.8 InfoListener Event

Triggered when the MobiLink server prints information.

Syntax

Visual Basic

```
Public Event InfoListener  As LogCallback
```

C#

```
public LogCallback InfoListener;
```

1.14.9 ShutdownListener Event

Triggered when the MobiLink server is shutting down.

Syntax

Visual Basic

```
Public Event ShutdownListener  As ShutdownCallback
```

C#

```
public ShutdownCallback ShutdownListener;
```

1.14.10 WarningListener Event

Triggered when the MobiLink server prints a warning.

Syntax

Visual Basic

```
Public Event WarningListener  As LogCallback
```

C#

```
public LogCallback WarningListener;
```

1.15 ServerException Class

Sends a signal to MobiLink when an error has occurred with the server to indicate that it should shut down immediately.

Syntax

Visual Basic

```
Public Class ServerException Inherits ScriptExecutionException
```

C#

```
public class ServerException : ScriptExecutionException
```

Members

All members of ServerException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	ServerException [page 91]	Creates a new ServerException with the default message.

Inherited members from ScriptExecutionException

Modifier and Type	Member	Description
public	ScriptExecutionException() [page 81]	Creates a new ScriptExecutionException with the default message.

Modifier and Type	Member	Description
public	ScriptExecutionException(string) [page 82]	Creates a new ScriptExecutionException with the given message.
protected	ScriptExecutionException(SerializationInfo, StreamingContext) [page 81]	Creates a new ScriptExecutionException from the given SerializationInfo and StreamingContext objects.
public	ScriptExecutionException(string, Exception) [page 82]	Creates a new ScriptExecutionException with the given message and containing the given inner exception that caused this one.

In this section:

ServerException Constructor [page 91]
Creates a new ServerException with the default message.

1.15.1 ServerException Constructor

Creates a new ServerException with the default message.

Overload list

Modifier and Type	Overload name	Description
public	ServerException() [page 92]	Creates a new ServerException with the default message.
protected	ServerException(SerializationInfo, StreamingContext) [page 92]	Creates a new ServerException from the given SerializationInfo and StreamingContext objects.
public	ServerException(string) [page 93]	Creates a new ServerException with the given message.
public	ServerException(string, Exception) [page 93]	Creates a new ServerException with the given message and containing the given inner exception that caused this one.

In this section:

ServerException() Constructor [page 92]
Creates a new ServerException with the default message.

ServerException(SerializationInfo, StreamingContext) Constructor [page 92]
Creates a new ServerException from the given SerializationInfo and StreamingContext objects.

ServerException(string) Constructor [page 93]
Creates a new ServerException with the given message.

ServerException(string, Exception) Constructor [page 93]
Creates a new ServerException with the given message and containing the given inner exception that caused this one.

1.15.1.1 ServerException() Constructor

Creates a new ServerException with the default message.

Syntax

Visual Basic

```
Public Sub ServerException ()
```

C#

```
public ServerException ()
```

1.15.1.2 ServerException(SerializationInfo, StreamingContext) Constructor

Creates a new ServerException from the given SerializationInfo and StreamingContext objects.

Syntax

Visual Basic

```
Protected Sub ServerException (
    ByVal info As SerializationInfo,
    ByVal context As StreamingContext
)
```

C#

```
protected ServerException (
    SerializationInfo info,
    StreamingContext context
)
```

Parameters

info The SerializationInfo object to construct this ServerException.
context The StreamingContext object to construct this ServerException.

1.15.1.3 ServerException(string) Constructor

Creates a new ServerException with the given message.

Syntax

Visual Basic

```
Public Sub ServerException (ByVal message As String)
```

C#

```
public ServerException (string message)
```

Parameters

message The message for this ServerException.

1.15.1.4 ServerException(string, Exception) Constructor

Creates a new ServerException with the given message and containing the given inner exception that caused this one.

Syntax

Visual Basic

```
Public Sub ServerException (
    ByVal message As String,
    ByVal ie As Exception
)
```

C#

```
public ServerException (
    string message,
    Exception ie
)
```

Parameters

message The message for this ServerException.
ie The exception that caused this ServerException.

1.16 SpatialUtilities Class

Represents a collection of static methods to work with spatial values.

Syntax

Visual Basic

```
Public NotInheritable Class SpatialUtilities
```

C#

```
public sealed class SpatialUtilities
```

Members

All members of SpatialUtilities, including inherited members.

Methods

Modifier and Type	Method	Description
public static byte[]	CreateSpatialValue(int, byte[]) [page 95]	Returns a new byte array that contains a spatial value formatted for download: the first four bytes contain the given SRID in little endian, and the remainder is the spatial data passed in the given byte array.
public static byte[]	GetBytes(byte[]) [page 95]	Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.
public static int	GetSRID(byte[]) [page 96]	Returns the SRID for the given spatial value.
public static void	SetSRID(byte[], int) [page 96]	Stores the given SRID in the first four bytes of the given byte array.

In this section:

CreateSpatialValue(int, byte[]) Method [page 95]
Returns a new byte array that contains a spatial value formatted for download: the first four bytes contain the given SRID in little endian, and the remainder is the spatial data passed in the given byte array.

GetBytes(byte[]) Method [page 95]
Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.

GetSRID(byte[]) Method [page 96]
Returns the SRID for the given spatial value.

SetSRID(byte[], int) Method [page 96]

Stores the given SRID in the first four bytes of the given byte array.

1.16.1 CreateSpatialValue(int, byte[]) Method

Returns a new byte array that contains a spatial value formatted for download: the first four bytes contain the given SRID in little endian, and the remainder is the spatial data passed in the given byte array.

Syntax

Visual Basic

```
Public Shared Function CreateSpatialValue (
    ByVal srid As Integer,
    ByVal spatial_value As Byte()
) As Byte()
```

C#

```
public static byte[] CreateSpatialValue (
    int srid,
    byte[] spatial_value
)
```

Parameters

srid The SRID.

spatial_value The spatial data.

Returns

The spatial value formatted for download.

1.16.2 GetBytes(byte[]) Method

Returns a new byte array with the same spatial data as the given byte array, but with the SRID removed.

Syntax

Visual Basic

```
Public Shared Function GetBytes (ByVal spatial_value As Byte()) As Byte()
```

C#

```
public static byte[] GetBytes (byte[] spatial_value)
```

Parameters

spatial_value A spatial value that needs its SRID removed

Returns

The new byte array.

1.16.3 GetSRID(byte[]) Method

Returns the SRID for the given spatial value.

Syntax

Visual Basic

```
Public Shared Function GetSRID (ByVal spatial_value As Byte()) As Integer
```

C#

```
public static int GetSRID (byte[] spatial_value)
```

Parameters

spatial_value The uploaded value. The first four bytes must contain the SRID encoded in little endian.

Returns

The SRID.

1.16.4 SetSRID(byte[], int) Method

Stores the given SRID in the first four bytes of the given byte array.

Syntax

Visual Basic

```
Public Shared Sub SetSRID (
```

```
    ByVal spatial_value As Byte(),
    ByVal srid As Integer
)
```

C#

```
public static void SetSRID (
    byte[] spatial_value,
    int srid
)
```

Parameters

spatial_value The array to store the SRID in.
srid The SRID to store.

1.17 SynchronizationException Class

Indicates when a synchronization exception has occurred and that the current synchronization should be rolled back and restarted.

Syntax

Visual Basic

```
Public Class SynchronizationException Inherits ScriptExecutionException
```

C#

```
public class SynchronizationException : ScriptExecutionException
```

Members

All members of SynchronizationException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
public	SynchronizationException [page 98]	Creates a new SynchronizationException with the default message.

Inherited members from ScriptExecutionException

Modifier and Type	Member	Description
public	ScriptExecutionException() [page 81]	Creates a new ScriptExecutionException with the default message.
public	ScriptExecutionException(string) [page 82]	Creates a new ScriptExecutionException with the given message.
protected	ScriptExecutionException(SerializationInfo, StreamingContext) [page 81]	Creates a new ScriptExecutionException from the given SerializationInfo and StreamingContext objects.
public	ScriptExecutionException(string, Exception) [page 82]	Creates a new ScriptExecutionException with the given message and containing the given inner exception that caused this one.

In this section:

SynchronizationException Constructor [page 98]
Creates a new SynchronizationException with the default message.

1.17.1 SynchronizationException Constructor

Creates a new SynchronizationException with the default message.

Overload list

Modifier and Type	Overload name	Description
public	SynchronizationException() [page 99]	Creates a new SynchronizationException with the default message.
protected	SynchronizationException(SerializationInfo, StreamingContext) [page 99]	Creates a new SynchronizationException from the given SerializationInfo and StreamingContext objects.
public	SynchronizationException(string) [page 100]	Creates a new SynchronizationException with the given message.
public	SynchronizationException(string, Exception) [page 100]	Creates a new SynchronizationException with the given message and containing the given inner exception that caused this one.

In this section:

SynchronizationException() Constructor [page 99]
Creates a new SynchronizationException with the default message.

SynchronizationException(SerializationInfo, StreamingContext) Constructor [page 99]

Creates a new SynchronizationException from the given SerializationInfo and StreamingContext objects.

SynchronizationException(string) Constructor [page 100]
Creates a new SynchronizationException with the given message.

SynchronizationException(string, Exception) Constructor [page 100]
Creates a new SynchronizationException with the given message and containing the given inner exception that caused this one.

1.17.1.1 SynchronizationException() Constructor

Creates a new SynchronizationException with the default message.

Syntax

Visual Basic

```
Public Sub SynchronizationException ()
```

C#

```
public SynchronizationException ()
```

1.17.1.2 SynchronizationException(SerializationInfo, StreamingContext) Constructor

Creates a new SynchronizationException from the given SerializationInfo and StreamingContext objects.

Syntax

Visual Basic

```
Protected Sub SynchronizationException (
    ByVal info As SerializationInfo,
    ByVal context As StreamingContext
)
```

C#

```
protected SynchronizationException (
    SerializationInfo info,
    StreamingContext context
)
```

Parameters

info The SerializationInfo object to construct this SynchronizationException.
context The StreamingContext object to construct this SynchronizationException.

1.17.1.3 SynchronizationException(string) Constructor

Creates a new SynchronizationException with the given message.

Syntax

Visual Basic

```
Public Sub SynchronizationException (ByVal message As String)
```

C#

```
public SynchronizationException (string message)
```

Parameters

message The message for this SynchronizationException.

1.17.1.4 SynchronizationException(string, Exception) Constructor

Creates a new SynchronizationException with the given message and containing the given inner exception that caused this one.

Syntax

Visual Basic

```
Public Sub SynchronizationException (
    ByVal message As String,
    ByVal ie As Exception
)
```

C#

```
public SynchronizationException (
    string message,
    Exception ie
)
```

Parameters

message The message for this SynchronizationException.
ie The exception that caused this SynchronizationException.

1.18 UpdateDataReader Interface

Holds the update operations for one upload transaction for one table.

Syntax

Visual Basic

```
Public Interface UpdateDataReader Implements System.Data.IDataReader
```

C#

```
public interface UpdateDataReader : System.Data.IDataReader
```

Members

All members of UpdateDataReader, including inherited members.

Methods

Modifier and Type	Method	Description
public void	SetNewRowValues() [page 102]	Sets the mode of this DataReader to return new column values (the post-updated row).
public void	SetOldRowValues() [page 102]	Sets the mode of this DataReader to return old column values (the pre-updated row).

Remarks

New and old rows can be accessed by changing the mode of the DataReader to old or new. This interface can otherwise be used as a regular DataReader.

In this section:

SetNewRowValues() Method [page 102]
Sets the mode of this DataReader to return new column values (the post-updated row).

SetOldRowValues() Method [page 102]
Sets the mode of this DataReader to return old column values (the pre-updated row).

1.18.1 SetNewRowValues() Method

Sets the mode of this DataReader to return new column values (the post-updated row).

Syntax

Visual Basic

```
Public Sub SetNewRowValues ()
```

C#

```
public void SetNewRowValues ()
```

Remarks

This is the default mode.

1.18.2 SetOldRowValues() Method

Sets the mode of this DataReader to return old column values (the pre-updated row).

Syntax

Visual Basic

```
Public Sub SetOldRowValues ()
```

C#

```
public void SetOldRowValues ()
```

1.19 UploadData Interface

Encapsulates upload operations for direct row handling.

Syntax

Visual Basic

```
Public Interface UploadData
```

C#

```
public interface UploadData
```

Members

All members of UploadData, including inherited members.

Methods

Modifier and Type	Method	Description
public UploadedTableData	GetUploadedTableByName(string) [page 104]	Gets the named Uploaded table data in this uploaded transaction.
public UploadedTableData[]	GetUploadedTables() [page 105]	Gets an array of all the uploaded tables data in this uploaded transaction.

Remarks

An upload transaction contains a set of tables containing row operations. An UploadData instance representing a single upload transaction is passed to the handle_UploadData synchronization event.

Note

You must handle direct row handling upload operations in the method registered for the handle_UploadData event. The UploadData is destroyed after each call to the registered method. Do not create a new instance of UploadData to use in subsequent events.

Use the UploadData.GetUploadedTables or UploadData.GetUploadedTableByName methods to obtain UploadedTableData instances.

A synchronization has one UploadData unless the remote database is using transactional or incremental upload.

Example

See handle_UploadData connection event.

In this section:

1.19.1 GetUploadedTableByName(string) Method

Gets the named Uploaded table data in this uploaded transaction.

Syntax

Visual Basic

```
Public Function GetUploadedTableByName (ByVal table_name As String) As
UploadedTableData
```

C#

```
public UploadedTableData GetUploadedTableByName (string table_name)
```

Parameters

table_name The name of the table for which we want the uploaded data

Returns

The uploaded data for the given table name, or null if not found.

Example

Assuming that you use a method called HandleUpload for the handle_UploadData synchronization event, the following example uses the GetUploadedTableByName method to return an UploadedTableData instance for the remoteOrders table:

```
// The method used for the handle_UploadData event.
```

```
public void HandleUpload(UploadData ut) {
   UploadedTableData uploaded_t1 = ut.GetUploadedTableByName("remoteOrders");
   // ...
}
```

1.19.2 GetUploadedTables() Method

Gets an array of all the uploaded tables data in this uploaded transaction.

Syntax

Visual Basic

```
Public Function GetUploadedTables () As UploadedTableData()
```

C#

```
public UploadedTableData[] GetUploadedTables ()
```

Returns

An array of uploaded table data. The order of tables in the array is the same as the upload order of the client.

Remarks

The order to the tables in the array is the same order that MobiLink uses for SQL row handling, and is the optimal order for preventing referential integrity violations. Use this table order if your data source is a relational database.

Example

Assuming that you use a method called HandleUpload for the handle_UploadData synchronization event, the following example uses the GetUploadedTables method to return UploadedTableData instances for the current upload transaction:

```
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ud) {
   UploadedTableData[] tables = ud.GetUploadedTables();
   //...
}
```

Related Information

UploadedTableData Interface [page 106]

1.20 UploadedTableData Interface

Encapsulates information for one uploaded table for a synchronization.

Syntax

Visual Basic

```
Public Interface UploadedTableData
```

C#

```
public interface UploadedTableData
```

Members

All members of UploadedTableData, including inherited members.

Methods

Modifier and Type	Method	Description
public IDataReader	GetDeletes() [page 107]	Gets a DataReader with the deletes for this uploaded table data.
public IDataReader	GetInserts() [page 108]	Gets a DataReader with the inserts for this uploaded table data.
public string	GetName() [page 109]	Gets the table name of this instance.
public DataTable	GetSchemaTable() [page 110]	Gets a DataTable that describes the metadata for this download table.
public UpdateDataReader	GetUpdates() [page 111]	Gets a DataReader with the updates for this uploaded table data.

Remarks

The insert, update and delete operations are all accessible via the standard ADO.NET IDataReader. The tables metadata can be accessed via the GetSchemaTable call or the insert and delete data readers. The delete data reader only includes the primary key columns of the table.

In this section:

1.20.1 GetDeletes() Method

Gets a DataReader with the deletes for this uploaded table data.

Syntax

Visual Basic

```
Public Function GetDeletes () As IDataReader
```

C#

```
public IDataReader GetDeletes ()
```

Returns

A DataReader with primary key columns for deleted rows.

Remarks

Each delete is represented by the primary key values needed to uniquely represent a row in this instances table.

> **Note**
>
> The index and order of the columns match the array for property DataTable.PrimaryKey for the schema of this table.

Example

Assuming that your remote client contains a table called sparse_pk, the following example uses the GetDeletes method to obtain a data reader of deleted rows. In this case, the delete DataReader includes two primary key columns. Note the index of each primary key column.

```
CREATE TABLE sparse_pk (
   pcol1  INT NOT NULL,
   col2 VARCHAR(200),
   pcol3 INT NOT NULL,
   PRIMARY KEY (pcol1, pcol3)
);
using Sap.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
   // Get an UploadedTableData for the sparse_pk table.
   UploadedTableData sparse_pk_table = ut.GetUploadedTableByName("sparse_pk");
   // Get deletes uploaded by the MobiLink client.
   using( IDataReader data_reader = sparse_pk_table.GetDeletes() ) {
            while (data_reader.Read()) {
           StringBuilder row_str = new StringBuilder("( ");
           row_str.Append(data_reader.GetString(0)); // pcol1
           row_str.Append(", ");
           row_str.Append(data_reader.GetString(1)); // pcol3
           row_str.Append(" )");
           writer.WriteLine(row_str);
            }
   }
}
```

1.20.2 GetInserts() Method

Gets a DataReader with the inserts for this uploaded table data.

Syntax

Visual Basic

```
Public Function GetInserts () As IDataReader
```

C#

```
public IDataReader GetInserts ()
```

Returns

A DataReader with inserts for this table data.

Remarks

Each insert is represented by one row in the result set.

Example

```
CREATE TABLE sparse_pk (
   pcol1  INT NOT NULL,
   col2 VARCHAR(200),
   pcol3 INT NOT NULL,
   PRIMARY KEY (pcol1, pcol3)
);
using Sap.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
   // Get an UploadedTableData for the sparse_pk table.
   UploadedTableData sparse_pk_table = ut.GetUploadedTableByName("sparse_pk");
   // Get inserts uploaded by the MobiLink client.
   using( IDataReader data_reader = sparse_pk_table.GetInserts() ) {
       while (data_reader.Read()) {
           StringBuilder row_str = new StringBuilder("( ");
           row_str.Append(data_reader.GetString(0)); // pcol1
           row_str.Append(", ");
           if (data_reader.IsDBNull(1)) {
               row_str.Append("&lt;NULL&gt;");
           }
           else {
               row_str.Append(data_reader.GetString(1)); // col2
           }
           row_str.Append(", ");
           row_str.Append(data_reader.GetString(2)); // pcol3
           row_str.Append(" )");
           writer.WriteLine(row_str);
       }
   }
}
```

1.20.3 GetName() Method

Gets the table name of this instance.

Syntax

Visual Basic

```
Public Function GetName () As String
```

C#

```
public string GetName ()
```

Returns

The table name of this instance.

Remarks

This is a utility function. The table name can also be accessed via the Schema for this instance.

1.20.4 GetSchemaTable() Method

Gets a DataTable that describes the metadata for this download table.

Syntax

Visual Basic

```
Public Function GetSchemaTable () As DataTable
```

C#

```
public DataTable GetSchemaTable ()
```

Returns

A DataTable that describes the column metadata.

Remarks

If you want the DataTable to contain column name information, you must specify the client option to send column names, which is the default behavior.

1.20.5 GetUpdates() Method

Gets a DataReader with the updates for this uploaded table data.

Syntax

Visual Basic

```
Public Function GetUpdates () As UpdateDataReader
```

C#

```
public UpdateDataReader GetUpdates ()
```

Returns

A DataReader with updates for this table data

Remarks

Each row in the result set represent one update. The mode of the result set can be flipped between new and old column values.

Example

The following example illustrates how to use the GetUpdates method:

```
CREATE TABLE sparse_pk (
   pcol1  INT NOT NULL,
   col2 VARCHAR(200),
   pcol3 INT NOT NULL,
   PRIMARY KEY (pcol1, pcol3)
);
using Sap.MobiLink.Script;
using System;
using System.IO;
using System.Data;
using System.Text;
...
// The method used for the handle_UploadData event.
public void HandleUpload(UploadData ut) {
   // Get an UploadedTableData for the sparse_pk table.
   UploadedTableData sparse_pk_table = ut.GetUploadedTableByName("sparse_pk");
   // Get updates uploaded by the MobiLink client.
   using( UpdateDataReader data_reader = sparse_pk_table.GetInserts() ) {
       while (data_reader.Read()) {
           data_reader.SetNewRowValues();
           StringBuilder row_str = new StringBuilder("New values ( ");
           row_str.Append(data_reader.GetString(0)); // pcol1
```

```
            row_str.Append(", ");
            if (data_reader.IsDBNull(1)) {
                row_str.Append("&lt;NULL&gt;");
            }
            else {
                row_str.Append(data_reader.GetString(1)); // col2
            }
            row_str.Append(", ");
            row_str.Append(data_reader.GetString(2)); // pcol3
            row_str.Append(" )");
            data_reader.SetOldRowValues();
            row_str.Append(" Old Values (");
            row_str.Append(data_reader.GetString(0)); // pcol1
            row_str.Append(", ");
            if (data_reader.IsDBNull(1)) {
                row_str.Append("&lt;NULL&gt;");
            }
            else {
                row_str.Append(data_reader.GetString(1)); // col2
            }
            row_str.Append(", ");
            row_str.Append(data_reader.GetString(2)); // pcol3
            row_str.Append(" )");
            writer.WriteLine(row_str);
        }
    }
}
```

1.21 LogCallback(ServerContext, LogMessage) Delegate

Called when the MobiLink server prints a message.

Syntax

Visual Basic

```
Public Delegate Sub LogCallback (
    ByVal sc As ServerContext,
    ByVal message As LogMessage
) As delegate void
```

C#

```
public delegate void LogCallback (
    ServerContext sc,
    LogMessage message
);
```

Remarks

Implementations of this delegate can be registered with the ServerContext events to be called when the MobiLink server prints a message.

Related Information

ErrorListener Event [page 89]
InfoListener Event [page 89]
WarningListener Event [page 90]

1.22 ShutdownCallback(ServerContext) Delegate

Called when MobiLink server is shutting down.

Syntax

Visual Basic

```
Public Delegate Sub ShutdownCallback (ByVal sc As ServerContext) As
delegate void
```

C#

```
public delegate void ShutdownCallback (ServerContext sc);
```

Remarks

Implementations of this delegate can be registered with the ShutdownListener event to be called when the MobiLink server shuts down.

Related Information

ShutdownListener Event [page 89]

1.23 SQLType Enumeration

Enumerates all possible ODBC data types.

Syntax

Visual Basic

```
Public Enum SQLType
```

C#

```
enum SQLType
```

Members

Member name	Description
SQL_TYPE_NULL	Null data type.
SQL_UNKNOWN_TYPE	Unknown data type.
SQL_CHAR	UTF-8 character array of a set size. Has .NET type String.
SQL_NUMERIC	Numeric value of set size and precision. Has .NET type Decimal.
SQL_DECIMAL	Decimal number of set size and precision. Has .NET type Decimal.
SQL_INTEGER	32-bit integer. Has .NET type Int32.
SQL_SMALLINT	16-bit integer. Has .NET type Int16
SQL_FLOAT	Floating point number with ODBC driver defined precision. Has .NET type Double.
SQL_REAL	Single precision floating point number. Has .NET type Single.
SQL_DOUBLE	Double precision floating point number. Has .NET type Double.
SQL_DATE	A date. Has .NET type DateTime.
SQL_DATETIME	A date and time. Has .NET type DateTime.
SQL_TIME	A time. Has .NET type DateTime
SQL_INTERVAL	An interval of time. Has .NET type TimeSpan

Member name	Description
SQL_TIMESTAMP	A time stamp. Has .NET type DateTime.
SQL_VARCHAR	A null terminated UTF-8 string with a user set maximum length. Has .NET type String.
SQL_TYPE_DATE	A date. Has .NET type DateTime.
SQL_TYPE_TIME	A time. Has .NET type DateTime.
SQL_TYPE_TIMESTAMP	A timestamp. Has .NET type DateTime
SQL_DEFAULT	A default type. Has no type.
SQL_ARD_TYPE	An ARD object. Has no type.
SQL_BIT	A single bit. Has .NET type Boolean.
SQL_TINYINT	8-bit integer. Has .NET type SByte.
SQL_BIGINT	64-bit integer. Has .NET type Int64.
SQL_LONGVARBINARY	Variable length binary data with a driver dependent maximum length. Has .NET type byte[].
SQL_VARBINARY	Variable length binary data with a user specified maximum length. Has .NET type byte[].
SQL_BINARY	Fixed length binary data. Has .NET type byte[].
SQL_LONGVARCHAR	A null terminated UTF-8 string with a driver dependent maximum length. Has .NET type String.
SQL_GUID	A GUID. Has .NET type Guid.

Member name	Description
SQL_WCHAR	Unicode character array of fixed size. Has .NET type String.
SQL_WVARCHAR	Null terminated Unicode string of user defined maximum length; Has .NET type String.
SQL_WLONGVARCHAR	Null terminated Unicode string of driver dependent maximum length; Has .NET type String.
SQL_SS_TIMESTAMPOFFSET	Timestamp with time zone offset; Has .NET type Sap.MobiLink.Script.DateTimeWithTimeZone This can be used only against Microsoft SQL Server and Oracle databases.

Remarks

Each SQLType corresponds to a .NET type.

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