ElevateDB Version 2 Data Access Components Manual

Table Of Contents

Chapter 1
Using the ODBC Driver

1.1 Application Compatibility

Supported Applications

The ElevateDB ODBC driver is an ODBC level 3 driver. We have tested the driver successfully with Microsoft Data Access Components (MDAC) version 2.7 or higher and the following applications:

Application	Versions and Notes
Crystal Reports	8.5 and later
Microsoft Office	2000 and later Microsoft Access has problems with using an auto-increment field as part of the primary index since the Jet engine cannot "discover" the keys properly when they are not populated explicitly by the client application.
Microsoft Visio	2000 and later
Borland Database Engine (BDE)	5.01 and later With the BDE there are problems with using an auto-increment field as part of the primary index since the BDE cannot "discover" the keys properly when they are not populated explicitly by the client application.
ADOExpress	Delphi 5 or later Only use a CursorLocation property of clUseClient. Server-side cursors do not work properly since the OLE layer deems dynamic cursors as not being capable of handling bookmark operations, even though such cursors can handle bookmark operations in ElevateDB.
ODBCExpress	5.06 and later
Microsoft ASP	5 and later It is recommended that you only use the ODBCDirect functionality in ASP and not the ADO->OLEDB->ODBC bridge driver through the ADO functionality. The bridge driver does not function correctly in most cases.
Microsoft Visual Basic	6 and later It is recommended that you only use the ODBCDirect functionality in VB 6 and not the ADO->OLEDB->ODBC bridge driver through the ADO functionality. The bridge driver does not function correctly in most cases.
Microsoft Visual Studio .NET	2002 and later

	It is recommended that you only use the ODBC.NET data provider with any .NET application (VB.NET, ASP.NET, C#, Delphi.NET, Chrome). Also, since the ODBC.NET data provider is accessing and using unmanaged resources and handles in the ODBC driver during operation, you should always call the Dispose method for any ODBCConnection, ODBCCommand, ODBCCommandBuilder, or ODBCDataAdapter objects when you are done using them (deterministic destruction). Failure to do so can cause major failures in the driver due to the resources and handles being freed up re-entrantly when the .NET garbage collector thread finalizes these objects.

Missing Features

There are still a few features and function calls missing from the driver, but they should not affect most environments. These missing features are:

- Support for bulk operations (SQLBulkOperations call)

- Support for working directly with descriptors (SQLSetDescRec, SQLGetDescRec, and SQLCopyDesc calls)

- Support for setting and getting cursor names (SQLGetCursorName and SQLSetCursorName calls)

More Information

The driver can completely handle all updating of data via SQL statements and the SQLExecute or SQLExecDirect calls, including BLOB data. Parameters are also completely supported, including BLOB parameters.

The driver provides scrollable cursor support via SQLFetchScroll and SQLExtendedFetch. The only two types of scrollable cursors supported are Static and Dynamic. Keyset-Driven cursors are not supported.

The driver cannot perform positioned updates using the SQL syntax WHERE CURRENT OF and using the SQLSetCursorName and SQLGetCursorName calls. This functionality is not supported in ElevateDB.

Even though the driver supports parameter arrays, you cannot request multiple result sets with the SQLMoreResults call. This is not supported in ElevateDB.

Any ODBC application that calls the SQLNumResultCols, SQLDescribeCol, or SQLColAttribute functions while a statement is prepared, but not executed, will force ElevateDB to execute the query in order to retrieve this result set information. ElevateDB does not support retrieving result set information until a SELECT statement is executed.

1.2 Data Source Configuration Tutorial

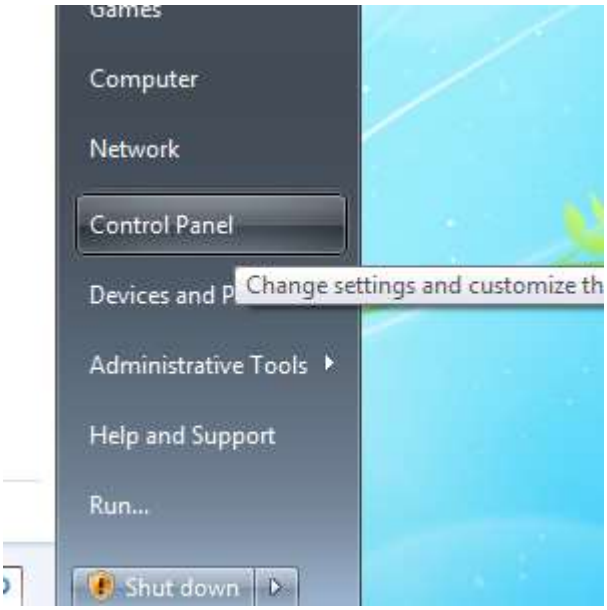
A data source, or DSN (Data Source Name), is used by applications that use ODBC to locate and access a specific database in a specific location. Once you have configured a data source in the ODBC Administrator, you may use this data source name in any application that can access ODBC. Please see the Application Compatibility topic in this manual for more information on applications that have been specifically tested with the ElevateDB ODBC Driver.

Step-By-Step Instructions

Complete the following steps to properly configure a data source that uses the ElevateDB ODBC Driver. These steps are illustrated using Windows 7, but are very similar for most other versions of Windows.

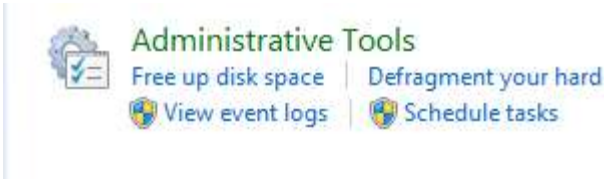
1. Run the **ODBC Administrator**, which is located in the **Administrative Tools** folder in the Control Panel. To reach the Administrative Tools folder, complete the following steps:

a. Click on the **Control Panel** link from the Start Menu.



b. Click on the **System and Security** link from the Start Menu.

d. Click on the **Administrative Tools** link in the System and Security window of the Control Panel.



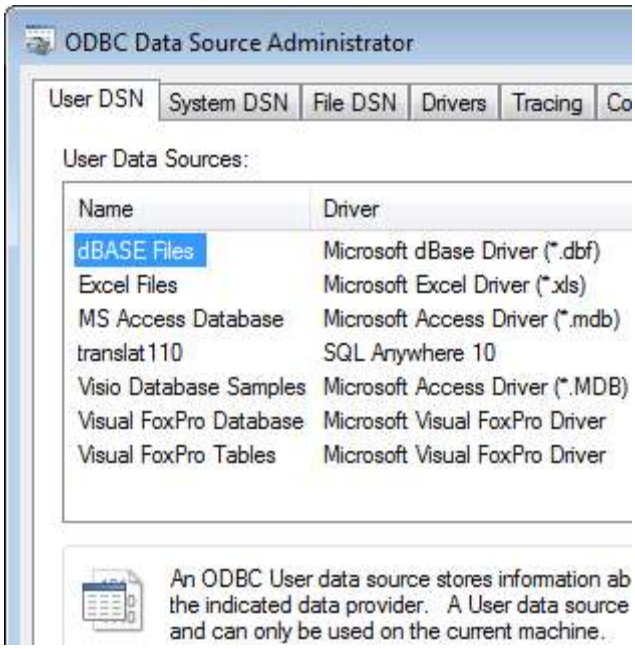
e. Double-click on the **Data Sources (ODBC)** icon in the Administrative Tools folder. This will bring forward the ODBC Administrator dialog.

> **Note**
> By default, 32-bit versions of Windows use the 32-bit ODBC Administrator, and 64-bit versions of Windows use the 64-bit ODBC Administrator, when launching the ODBC Administrator via the Administrative Tools link. If you're using a 64-bit version of Windows, then you must use the 32-bit ODBC Administrator located here in order to configure 32-bit data sources for use with 32-bit applications and the 32-bit ElevateDB ODBC Driver:

```
<WindowsInstallDir>\SysWoW64\odbcad32.exe
```

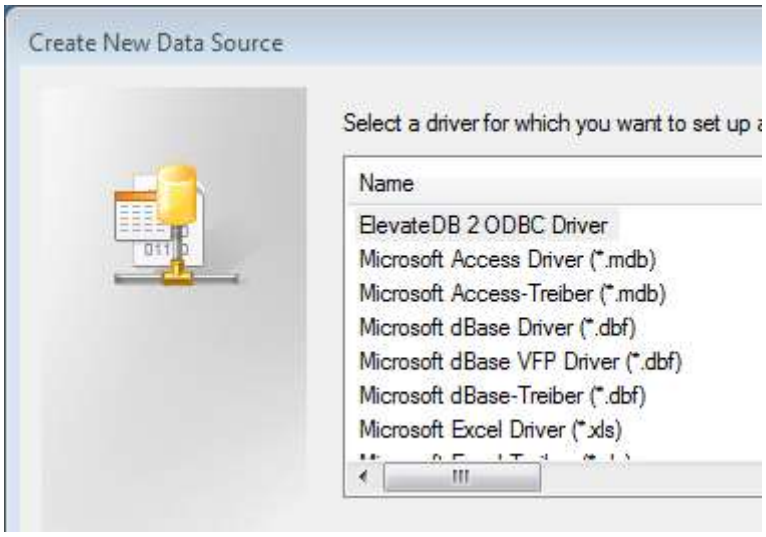
where <WindowsInstallDir> is the base Windows installation directory, usually c:\Windows.

2. If you want the data source to be accessible by all users, then click on the **System DSN** tab. If you want the data source to only be accessible to the current user, then click **User DSN** tab (the default page). For the rest of this tutorial, we will be configuring a new User DSN.



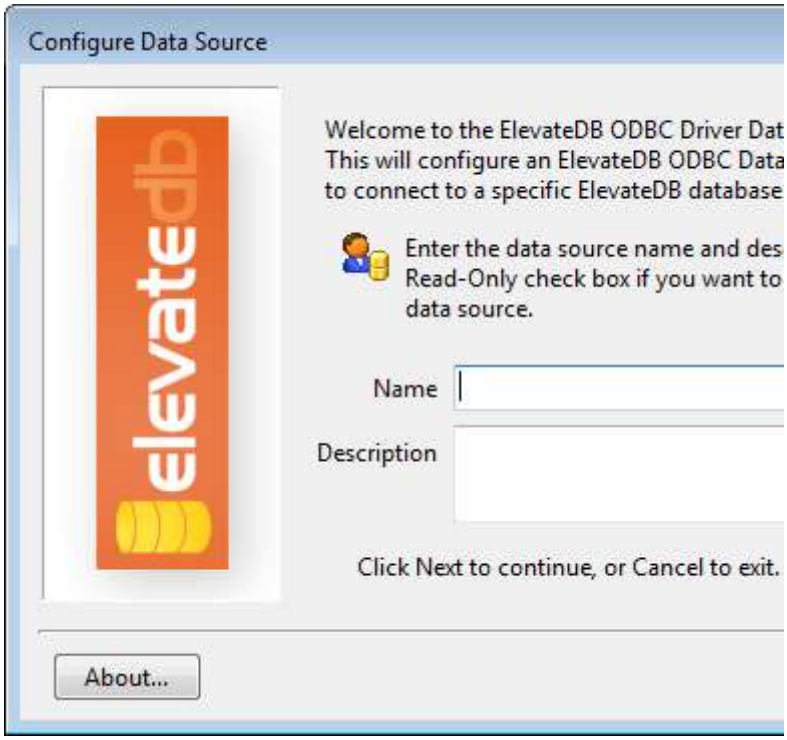
Note
Only Administrators can define System DSNs since they require access to the HKEY_LOCAL_MACHINE registry hive of the registry.

3. Click on the **Add** button to begin adding a data source. This will bring forward a dialog with a list of the installed ODBC drivers. Select the **ElevateDB ODBC Driver** from the list of drivers.



5. Click on the **Finish** button to complete the driver selection process and begin the configuration process.

6. An ElevateDB ODBC Driver configuration wizard dialog will now be shown. Follow the instructions on this wizard to complete the data source configuration.



7. Once the configuration steps have been completed and you have clicked on the Finish button in the dialog, you will be brought back to the DSN page of the ODBC Administrator where you should now see the DSN that you have just created.

If at any time you wish to re-configure the data source, simply choose the appropriate tab page (User DSN or System DSN), select the data source name that you previously added from the list of data sources, and then click on the Configure button. This will bring forward the same configuration wizard dialog as before, except in this case you cannot specify a name for the data source.

1.3 Registry Entries

Location

ODBC data sources are stored in the registry in Windows. The location of both user and system data sources is detailed below:

Data Source Type	Location
User DSN	HKEY_CURRENT_USER\Software\ODBC\ODBC.INI\<Data Source Name>
System DSN	HKEY_LOCAL_MACHINE\Software\ODBC\ODBC.INI\<Data Source Name>

Also, a list of data sources defined on the system can be found here:

Data Source Type	Location
User DSN	HKEY_CURRENT_USER\Software\ODBC\ODBC.INI\ODBC Data Sources
System DSN	HKEY_LOCAL_MACHINE\Software\ODBC\ODBC.INI\ODBC Data Sources

64-bit Windows

Under 64-bit Windows, the above registry keys/values are for 64-bit DSNs only, and are only configurable via the 64-bit ODBC Administrator that is accessible from the Control Panel. In order to configure 32-bit DSNs on 64-bit Windows, one must use the 32-bit ODBC Administrator located here:

```
<WindowsInstallDir>\SysWoW64\odbcad32.exe
```

where <WindowsInstallDir> is the base Windows installation directory, usually c:\Windows.

In addition, 32-bit ODBC Administrator uses the special 32-bit registry values here for data sources:

Data Source Type	Location
User DSN	HKEY_CURRENT_USER\Software\Wow6432Node\ODBC\ODBC.INI\ODBC Data Sources
System DSN	HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBC.INI\ODBC Data Sources

and here for a list of data sources:

Data Source Type	Location

User DSN	HKEY_CURRENT_USER\Software\Wow6432Node\ODBC\ODBC.INI\<Data Source Name>
System DSN	HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBC.INI\<Data Source Name>

ElevateDB Data Source Settings

The following registry values are defined under the <Data Source Name> key in the registry (see above). All registry values marked with the (R) symbol next to their name are only applicable when the TYPE registry value is set to "REMOTE". Likewise, all registry values marked with the (L) symbol next to their name are only applicable when the TYPE registry value is set to "LOCAL".

Name	Description
DRIVER	This string value is always set to the directory and file name of the ElevateDB ODBC Driver for which the data source is configured.
NAME	This string value specifies the name of connection, and is the same as the DSN.
DESCRIPTION	This string value specifies the description of the connection.
CHARSET	This string value specifies which character set, "ANSI" or "UNICODE", to use for the connection. For remote connections to an ElevateDB Server, the value must match the character set being used by the ElevateDB Server. The default value is "UNICODE".
TYPE	This string value is set to either "LOCAL" if the data source is accessing the database directly, or "REMOTE" if the data source is accessing the database remotely via an ElevateDB Server.
HOST (R)	This string value specifies the host name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS registry values must be populated along with the SERVICE or PORT registry values in order to correctly access a database on an ElevateDB Server. The default value is "".
ADDRESS (R)	This string value specifies the IP address of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS registry values must be populated along with the SERVICE or PORT registry values in order to correctly access a database on an ElevateDB Server. The default value is "127.0.0.1".
SERVICE (R)	This string value specifies the service name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS registry values must be populated along with the SERVICE or PORT registry values in order to correctly access a database on an ElevateDB Server. The default value is "".
PORT (R)	This string value specifies the port number of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS registry values must be populated along with the SERVICE or PORT registry values in order to correctly access a database on an ElevateDB Server. The default value is

	"12010".
PING (R)	This string value specifies whether pinging will be enabled for the connection to the ElevateDB Server. When pinging is enabled, the driver will send a keep-alive request to the ElevateDB Server in the interval specified by the PINGINTERVAL registry value for the data source. This prevents the ElevateDB Server from disconnecting and/or removing the connection, even if the connection has been idle for a very long period of time. Please see the Server Session Timeout configuration item in the Starting and Configuring the ElevateDB Server topic for more information on how idle connections are handled. Specify "TRUE" to enable pinging, or "FALSE" (the default) to disable pinging.
PINGINTERVAL (R)	This string value specifies how often the connection will ping the ElevateDB Server when pinging is enabled via the PING registry value for the data source. The default value is "60" (seconds).
TIMEOUT (R)	This string value specifies how long the connection will wait on a response from the ElevateDB Server before disconnecting and issuing an error. The default value is "180" (seconds).
ENCRYPTED (R)	This string value specifies whether the connection to the ElevateDB Server should be encrypted or no. Specify "TRUE" to enable encryption for the connection, or "FALSE" (the default) to disable encryption for the connection.
ENCRYPTSRVPWD (R)	This string value specifies the password to use for encrypted connections to the remote ElevateDB Server, as well as for encrypting logins to the remote ElevateDB Server. This password must match the configured encryption password for the remote ElevateDB Server, and you should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".
COMPRESSION (R)	This string value specifies the amount of compression to use when communicating with the ElevateDB Server. The default value is "0", or no compression. A value of "1" to "10" specifies the amount of compression from fast, but not very thorough, to very thorough, but not as fast. A value of "6" specifies a balance between size and speed, and represents the ideal level for most applications. **Note** The ElevateDB ODBC Driver will automatically adjust this value for situations where the existing compression level is not ideal, such as in cases where the amount of data being sent is so small that it is of no benefit to compress the data.
READAHEADROWS (R)	This string value specifies how many rows will be read in a single request when the connection is requesting rows from the remote ElevateDB Server. The default value is "10".

CONFIGMEMORY (L)	This string value specifies whether the configuration file used by the data source will be located in the process memory ("virtual") or on disk. Specify "TRUE" to use a virtual configuration file, or "FALSE" (the default) to use a disk-based configuration file in the location specified by the CONFIGPATH registry value (see below). The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on configuration files.
CONFIGPATH (L)	This string value specifies the configuration path to use for the data source. The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on the configuration path.
TEMPPATH (L)	This string value specifies the temporary tables path to use for the data source. The temporary tables path is used to store any temporary tables generated during query execution. The default value is the operating system setting for the storing temporary files for the current user. Please see the Architecture for more information on the temporary tables path.
KEEPTABLESOPEN	This string value specifies whether tables should be kept open for the duration of the connection once they have been opened at least once. Specify "TRUE" to keep tables open, or "FALSE" (the default) to have tables opened and closed on demand. Setting this value to "TRUE" can result in improved performance, especially with applications that execute many singleton SQL statements in a row.
CONFIGNAME (L)	This string value specifies the root name (without extension) used by the data source for the configuration file. The default value is "EDBConfig". The extension used for the configuration file is determined by the CONFIGEXT value. The location of the configuration file is determined by the CONFIGPATH value.
CONFIGEXT (L)	This string value specifies the extension used by the data source for the configuration file. The default value is ".EDBCfg". The root name (without extension) used for the configuration file is determined by the CONFIGNAME value. The location of the configuration file is determined by the CONFIGPATH value.
LOCKEXT (L)	This string value specifies the extension used by the data source for both the configuration and database catalog lock files. The default value is ".EDBLck". The root name (without extension) used for the configuration lock file is determined by the CONFIGNAME value. The root name (without extension) used for a database catalog lock file is determined by the CATALOGNAME value. The location of the configuration lock file is determined by the CONFIGPATH value, and the configuration lock file is hidden, by default. The location of a database catalog lock file is determined by the path designated for the applicable database when the database was created, and a database catalog lock file is hidden, by

	default.
LOGEXT (L)	This string value specifies the extension used by the data source for the configuration log file. The default value is ".EDBLog". The root name (without extension) used for the configuration log file is determined by the CONFIGNAME value. The location of the configuration log file is determined by the CONFIGPATH value.
MAXLOGSIZE (L)	This string value specifies the maximum size of the log file (in bytes) that the log file can grow to. Log entries are added to the log in a circular fashion, meaning that once the maximum log file size ia reached, ElevateDB will start re-using the oldest log entries for new log entries. The default value is 1048576 bytes. Which types of logged events are recorded in the log can be controlled by the LOGCATS value. By default, all categories of events are logged (INFO, WARN, and ERROR). **Warning** It is very important that all data sources and/or applications accessing the same configuration file use the same maximum log file size for the configuration log file. Using different values can result in log entries being prematurely overwritten or appearing "out-of-order" when viewing the log entries via the LogEvents Table.
LOGCATS (L)	This string value specifies the types of events that should be logged in the configuration log file for the current data source, with each value separated by a comma (,). The default value is "INFO,WARN,ERROR", or all categories of events.
CATALOGNAME (L)	This string value specifies the root name (without extension) used by the data source for all database catalog files. The default value is "EDBDatabase". The extension used for the catalog files is determined by the CATALOGEXT value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
CATALOGEXT (L)	This string value specifies the extension used by the data source for database catalog files. The default value is ".EDBCat". The root name (without extension) used for all database catalog files is determined by the CATALOGNAME value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
BACKUPEXT (L)	This string value specifies the extension used by the data source for database backup files. The default value is ".EDBBkp". The root name (without extension) used for a database backup file is determined by the name given when the BACKUP DATABASE statement is executed.

UPDATEEXT (L)	This string value specifies the extension used by the data source for database update files. The default value is ".EDBUpd". The root name (without extension) used for a database update file is determined by the name given when the SAVE UPDATES statement is executed.
TABLEEXT (L)	This string value specifies the extension used by the data source for database table files. The default value is ".EDBTbl". The root name (without extension) used for a table file is determined by the name given when the CREATE TABLE statement is executed. The location of the table files is determined by the path designated for the applicable database when the database was created.
INDEXEXT (L)	This string value specifies the extension used by the data source for database table index files. The default value is ".EDBIdx". The root name (without extension) used for a table's index file is determined by the name given when the CREATE TABLE statement is executed. The location of the table index files is determined by the path designated for the applicable database when the database was created.
BLOBEXT (L)	This string value specifies the extension used by the data source for database table BLOB files. The default value is ".EDBBlb". The root name (without extension) used for a table's BLOB file is determined by the name given when the CREATE TABLE statement is executed. The location of the table BLOB files is determined by the path designated for the applicable database when the database was created.
PUBLISHEXT (L)	This string value specifies the extension used by the data source for database table publish files. The default value is ".EDBPbl". The root name (without extension) used for a table's publish file is determined by the name given when the CREATE TABLE statement is executed. The location of the table publish files is determined by the path designated for the applicable database when the database was created.
UID	This string value specifies the user ID to use when connecting to the data source. If this value is left blank, the user will be prompted for the user ID, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
PWD	This string value specifies the password to use when connecting to the data source. If this value is left blank, the user will be prompted for the password, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
DATABASE	This string value specifies the name of the database being accessed with the data source. Please see the Architecture for more information on databases.

READONLY	This string value specifies whether the data source is read-only or read-write. Set this value to "TRUE" to make the data source read-only, and "FALSE" to make the data source read-write.
ROWLOCKPROTOCOL	This string value specifies which row locking protocol to use for the data source. Set this value to "PESSIMISTIC" to specify that all UPDATE row locks should be acquired when the rows are read during the process of the UPDATEs. Set this value to "OPTIMISTIC" to specify that all UPDATE row locks should only be acquired when the rows are actually being updated. Please see the Locking and Concurrency topic for more information.
ROWLOCKRETRIES	This string value specifies the number of row lock attempts that the driver should make before issuing an error. The default is "15".
ROWLOCKWAIT	This string value specifies the amount of time (in milliseconds) to wait between each row lock attempt. The default is "100".
DETECTROWCHANGES	This string value specifies whether the driver should issue an error when a row is updated and the row has changed since it was last cached. Specify "TRUE" to enable row change detection, or "FALSE" (the default) to disable row change detection. Please see the Change Detection topic for more information.
CATALOGINFO (L)	This string value specifies whether database catalog character set and version information should appear in the Databases system information table. The default value is "TRUE". **Note** Setting the value to "FALSE" can significantly improve the performance of the loading of the Databases system information table when there are a lot of databases in a configuration. This is because ElevateDB has to open the database catalog for each database in order to read the character set and version number.
CACHEMODULES (L)	This string value specifies whether module DLLs should be cached in memory for the duration of the connection. The default value is "FALSE".
STMTCACHESIZE	This string value specifies how many SQL statements can be cached in memory for the duration of the connection. Caching SQL statements improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that SQL statements will not be cached for the connection.

	Note The maximum number of open SQL statements per connection is 2048, so you should not set the statement cache size that high. Also, the SQL statement cache size is a **per-open-database** setting.
PROCCACHESIZE	This string value specifies how many functions/procedures can be cached in memory for the duration of the connection. Caching functions/procedures improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that functions/procedures will not be cached for the connection. **Note** The maximum number of open functions/procedures per connection is 2048, so you should not set the procedure cache size that high. Also, the function/procedure cache size is a **per-open-database** setting.
FLUSHWRITES	This string value controls whether the driver forces the operating system to flush any buffered writes to disk immediately after the data is written to the operating system. If the value is "FALSE" (the default), then ElevateDB will leave the flushing up to the operating system. If it is "TRUE", then ElevateDB will force a buffer flush after every write. Please see the Buffering and Caching topic for more information.
SIGNATURE	This string value specifies the signature to use for the data source. A signature is used to "sign" all configuration and database files created by ElevateDB so that they are only accessible using that signature, as well as "signing" all communications with a remote ElevateDB Server. You should only specify this value if you know exactly what you are doing and need to use a different signature than the default of "edb_signature".
ENCRYPTPWD (L)	This string value specifies the password to use for encrypting any local engine files. You should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".

1.4 Connection Strings

Connection strings are used when the SQLDriverConnect and SQLBrowseConnect ODBC API functions are called. They may specify as little as a data source name or as much as an entire data source configuration. The SQLDriverConnect function will interactively complete a connection string, if necessary, and, if the calling program indicates that it wants this behavior, by prompting the user for the missing information. On the other hand, the SQLBrowseConnect function will do so programmatically by iteratively interacting with the calling program. For more information on the SQLDriverConnect and SQLBrowseConnect API calls, please refer to the ODBC Programmers Reference from Microsoft. For more information on what function calls are used in your application program, please ask the vendor of the application program being used.

Pre-Configured Data Source Connection Strings

Connection strings that connect to a pre-configured data source, do so by specifying the DSN keyword in the connection string:

```
DSN=MyDataSource
```

Any other keywords that are specified in the connection string are overridden with the settings present in the data source configuration for the specified data source.

In addition, you can also use the FILEDSN keyword to load a data source configuration from a specific file instead of the registry:

```
FILEDSN=c:\windows\temp\mydatasource.dsn
```

That will load the configuration values from the mydatasource.dsn file. For more information on using the FILEDSN keyword, please refer to the ODBC Programmers Reference from Microsoft.

Direct Connection Strings

Direct connection strings bypass a pre-configured data source altogether and specify all of the keywords necessary to configure and access a given data source. The first keyword in a direct connection string must always be the special DRIVER keyword. For the ElevateDB ODBC Driver, it would look like this:

```
DRIVER={ElevateDB 2 ODBC Driver}
```

Notice the use of the required braces {} around the DRIVER keyword.

The keywords that can be used with direct connection strings and the ElevateDB ODBC Driver are listed below. Here is an example direct connection string that connects to a remote ElevateDB Server and a database called "Accounting" (case-insensitive):

```
DRIVER={ElevateDB 2 ODBC Driver};
CHARSET=UNICODE;
```

```
TYPE=REMOTE;
ADDRESS=192.168.0.28;
DATABASE=Accounting
```

Note
The line breaks inserted above are only for readability and should not be used in an actual connection string.

Connection String Keywords

The following keywords are used with connection strings. All keywords marked with the (R) symbol next to their name are only applicable when the TYPE keyword is set to "REMOTE". Likewise, all keywords marked with the (L) symbol next to their name are only applicable when the TYPE keyword is set to "LOCAL".

Name	Description
CHARSET	This string value specifies which character set, "ANSI" or "UNICODE", to use for the connection. For remote connections to an ElevateDB Server, the value must match the character set being used by the ElevateDB Server. The default value is "UNICODE".
TYPE	This string value is set to either "LOCAL" if the connection is accessing the database directly, or "REMOTE" if the connection is accessing the database remotely via an ElevateDB Server.
NAME	This string value specifies the name of the connection.
DESCRIPTION	This string value specifies the description of the connection.
HOST (R)	This string value specifies the host name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".
ADDRESS (R)	This string value specifies the IP address of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "127.0.0.1".
SERVICE (R)	This string value specifies the service name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".
PORT (R)	This string value specifies the port number of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "12010".

CONNECTTIMEOUT (R)	This string value specifies the maximum amount of time, in seconds, that ElevateDB will wait for a successful connection before aborting the connection attempt. The default value is "15" (seconds).
PING (R)	This string value specifies whether pinging will be enabled for the connection to the ElevateDB Server. When pinging is enabled, the driver will send a keep-alive request to the ElevateDB Server in the interval specified by the PINGINTERVAL value. This prevents the ElevateDB Server from disconnecting and/or removing the connection, even if the connection has been idle for a very long period of time. Please see the Server Session Timeout configuration item in the Starting and Configuring the ElevateDB Server topic for more information on how idle connections are handled. Specify "TRUE" to enable pinging, or "FALSE" (the default) to disable pinging.
PINGINTERVAL (R)	This string value specifies how often the connection will ping the ElevateDB Server when pinging is enabled via the PING value. The default value is "60" (seconds).
TIMEOUT (R)	This string value specifies how long the connection will wait on a response from the ElevateDB Server before disconnecting and issuing an error. The default value is "180" (seconds).
ENCRYPTED (R)	This string value specifies whether the connection to the ElevateDB Server should be encrypted or no. Specify "TRUE" to enable encryption for the connection, or "FALSE" (the default) to disable encryption for the connection.
ENCRYPTSRVPWD (R)	This string value specifies the password to use for encrypted connections to the remote ElevateDB Server, as well as for encrypting logins to the remote ElevateDB Server. This password must match the configured encryption password for the remote ElevateDB Server, and you should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".
COMPRESSION (R)	This string value specifies the amount of compression to use when communicating with the ElevateDB Server. The default value is "0", or no compression. A value of "1" to "10" specifies the amount of compression from fast, but not very thorough, to very thorough, but not as fast. A value of "6" specifies a balance between size and speed, and represents the ideal level for most applications. **Note** The ElevateDB ODBC Driver will automatically adjust this value for situations where the existing compression level is not ideal, such as in cases where the amount of data being sent is so small that it is of no benefit to compress the data.

READAHEADROWS (R)	This string value specifies how many rows will be read in a single request when the connection is requesting rows from the remote ElevateDB Server. The default value is "10".
CONFIGMEMORY (L)	This string value specifies whether the configuration file used by the connection will be located in the process memory ("virtual") or on disk. Specify "TRUE" to use a virtual configuration file, or "FALSE" (the default) to use a disk-based configuration file in the location specified by the CONFIGPATH string value (see below). The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on configuration files.
CONFIGPATH (L)	This string value specifies the configuration path to use for the connection. The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on the configuration path.
TEMPPATH (L)	This string value specifies the temporary tables path to use for the connection. The temporary tables path is used to store any temporary tables generated during query execution. The default value is the operating system setting for the storing temporary files for the current user. Please see the Architecture for more information on the temporary tables path.
KEEPTABLESOPEN	This string value specifies whether tables should be kept open for the duration of the connection once they have been opened at least once. Specify "TRUE" to keep tables open, or "FALSE" (the default) to have tables opened and closed on demand. Setting this value to "TRUE" can result in improved performance, especially with applications that execute many singleton SQL statements in a row.
CONFIGNAME (L)	This string value specifies the root name (without extension) used by the connection for the configuration file. The default value is "EDBConfig". The extension used for the configuration file is determined by the CONFIGEXT value. The location of the configuration file is determined by the CONFIGPATH value.
CONFIGEXT (L)	This string value specifies the extension used by the connection for the configuration file. The default value is ".EDBCfg". The root name (without extension) used for the configuration file is determined by the CONFIGNAME value. The location of the configuration file is determined by the CONFIGPATH value.
LOCKEXT (L)	This string value specifies the extension used by the connection for both the configuration and database catalog lock files. The default value is ".EDBLck". The root name (without extension) used for the configuration lock file is determined by the CONFIGNAME value. The root name (without extension) used for a database catalog lock file is determined by the CATALOGNAME value. The location of the configuration lock file is determined by the CONFIGPATH value, and the configuration lock file is hidden, by default.

	The location of a database catalog lock file is determined by the path designated for the applicable database when the database was created, and a database catalog lock file is hidden, by default.
LOGEXT (L)	This string value specifies the extension used by the connection for the configuration log file. The default value is ".EDBLog". The root name (without extension) used for the configuration log file is determined by the CONFIGNAME value. The location of the configuration log file is determined by the CONFIGPATH value.
MAXLOGSIZE (L)	This string value specifies the maximum size of the log file (in bytes) that the log file can grow to. Log entries are added to the log in a circular fashion, meaning that once the maximum log file size ia reached, ElevateDB will start re-using the oldest log entries for new log entries. The default value is 1048576 bytes. Which types of logged events are recorded in the log can be controlled by the LOGCATS value. By default, all categories of events are logged (INFO, WARN, and ERROR). **Warning** It is very important that all data sources and/or applications accessing the same configuration file use the same maximum log file size for the configuration log file. Using different values can result in log entries being prematurely overwritten or appearing "out-of-order" when viewing the log entries via the LogEvents Table.
LOGCATS (L)	This string value specifies the types of events that should be logged in the configuration log file for the current connection, with each value separated by a comma (,). The default value is "INFO,WARN,ERROR", or all categories of events.
CATALOGNAME (L)	This string value specifies the root name (without extension) used by the connection for all database catalog files. The default value is "EDBDatabase". The extension used for the catalog files is determined by the CATALOGEXT value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
CATALOGEXT (L)	This string value specifies the extension used by the connection for database catalog files. The default value is ".EDBCat". The root name (without extension) used for all database catalog files is determined by the CATALOGNAME value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
BACKUPEXT (L)	This string value specifies the extension used by the connection for database backup files. The default value is ".EDBBkp". The root name (without extension) used for a database backup file is determined by the name given when the BACKUP DATABASE statement is executed.

UPDATEEXT (L)	This string value specifies the extension used by the connection for database update files. The default value is ".EDBUpd". The root name (without extension) used for a database update file is determined by the name given when the SAVE UPDATES statement is executed.
TABLEEXT (L)	This string value specifies the extension used by the connection for database table files. The default value is ".EDBTbl". The root name (without extension) used for a table file is determined by the name given when the CREATE TABLE statement is executed. The location of the table files is determined by the path designated for the applicable database when the database was created.
INDEXEXT (L)	This string value specifies the extension used by the connection for database table index files. The default value is ".EDBIdx". The root name (without extension) used for a table's index file is determined by the name given when the CREATE TABLE statement is executed. The location of the table index files is determined by the path designated for the applicable database when the database was created.
BLOBEXT (L)	This string value specifies the extension used by the connection for database table BLOB files. The default value is ".EDBBlb". The root name (without extension) used for a table's BLOB file is determined by the name given when the CREATE TABLE statement is executed. The location of the table BLOB files is determined by the path designated for the applicable database when the database was created.
PUBLISHEXT (L)	This string value specifies the extension used by the connection for database table publish files. The default value is ".EDBPbl". The root name (without extension) used for a table's publish file is determined by the name given when the CREATE TABLE statement is executed. The location of the table publish files is determined by the path designated for the applicable database when the database was created.
UID	This string value specifies the user ID to use when connecting to the connection. If this value is left blank, the user will be prompted for the user ID, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
PWD	This string value specifies the password to use when connecting to the connection. If this value is left blank, the user will be prompted for the password, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
DATABASE	This string value specifies the name of the database being accessed with the connection. Please see the Architecture for more information on databases.

READONLY	This string value specifies whether the connection is read-only or read-write. Set this value to "TRUE" to make the connection read-only, and "FALSE" to make the connection read-write.
ROWLOCKPROTOCOL	This string value specifies which row locking protocol to use. Set this value to "PESSIMISTIC" to specify that all UPDATE row locks should be acquired when the rows are read during the process of the UPDATEs. Set this value to "OPTIMISTIC" to specify that all UPDATE row locks should only be acquired when the rows are actually being updated. Please see the Locking and Concurrency topic for more information.
ROWLOCKRETRIES	This string value specifies the number of row lock attempts that the driver should make before issuing an error. The default is "15".
ROWLOCKWAIT	This string value specifies the amount of time (in milliseconds) to wait between each row lock attempt. The default is "100".
DETECTROWCHANGES	This string value specifies whether the driver should issue an error when a row is updated and the row has changed since it was last cached. Specify "TRUE" to enable row change detection, or "FALSE" (the default) to disable row change detection. Please see the Change Detection topic for more information.
CATALOGINFO (L)	This string value specifies whether database catalog character set and version information should appear in the Databases system information table. The default value is "TRUE". **Note** Setting the value to "FALSE" can significantly improve the performance of the loading of the Databases system information table when there are a lot of databases in a configuration. This is because ElevateDB has to open the database catalog for each database in order to read the character set and version number.
CACHEMODULES (L)	This string value specifies whether module DLLs should be cached in memory for the duration of the connection. The default value is "FALSE".
STMTCACHESIZE	This string value specifies how many SQL statements can be cached in memory for the duration of the connection. Caching SQL statements improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that SQL statements will not be cached for the connection.

	Note The maximum number of open SQL statements per connection is 2048, so you should not set the statement cache size that high. Also, the SQL statement cache size is a **per-open-database** setting.
PROCCACHESIZE	This string value specifies how many functions/procedures can be cached in memory for the duration of the connection. Caching functions/procedures improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that functions/procedures will not be cached for the connection. **Note** The maximum number of open functions/procedures per connection is 2048, so you should not set the procedure cache size that high. Also, the function/procedure cache size is a **per-open-database** setting.
FLUSHWRITES	This string value controls whether the driver forces the operating system to flush any buffered writes to disk immediately after the data is written to the operating system. If the value is "FALSE" (the default), then ElevateDB will leave the flushing up to the operating system. If it is "TRUE", then ElevateDB will force a buffer flush after every write. Please see the Buffering and Caching topic for more information.
SIGNATURE	This string value specifies the signature to use for the connection. A signature is used to "sign" all configuration and database files created by ElevateDB so that they are only accessible using that signature, as well as "signing" all communications with a remote ElevateDB Server. You should only specify this value if you know exactly what you are doing and need to use a different signature than the default of "edb_signature".
ENCRYPTPWD (L)	This string value specifies the password to use for encrypting any local engine files. You should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".

1.5 Custom Driver Installation

Location

ODBC drivers are installed and configured using the registry in Windows. The location of the driver entries is the following registry key:

```
HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI\<Driver Name>
```

In addition, the name of the driver must also be added to the following registry key:

```
HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI\ODBC Drivers
```

The name of the registry value is the name of the ODBC driver, and the data for the registry value is a string with the value "Installed" (without surrounding double quotes). For example, for the ElevateDB 2 ODBC Driver, the entire registry key and value would be the following:

```
Key: HKEY_LOCAL_MACHINE\Software\ODBC\ODBCINST.INI\ODBC Drivers

Value Name: ElevateDB 2 ODBC Driver
Value Type: STRING
Value Data: Installed
```

64-bit Windows

Under 64-bit Windows, the above registry keys/values are for 64-bit drivers only. In order to configure 32-bit drivers on 64-bit Windows, one must use the following registry key instead:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBCINST.INI\<Driver Name>
```

In addition, the name of the driver must also be added to the following registry key:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBCINST.INI\ODBC Drivers
```

ElevateDB ODBC Driver Settings

The following registry values are defined under the <Driver Name> key in the registry (see above). These registry settings are all required, and should be specified exactly as indicated in order to ensure proper operation.

Value Name	Type and Description

APILevel	STRING This value should always be set to "1" (without surrounding double quotes).
ConnectFunctions	STRING This value should always be set to "YYY" (without surrounding double quotes).
Driver	STRING This value should always be set to the location of the ODBC driver DLL (edbodbc.dll, by default). This location can be anywhere on a local machine drive.
DriverODBCVer	STRING This value should always be set to "03.00" (without surrounding double quotes).
FileExtns	STRING This value should always be set to "*.EDBTbl,*.EDBIdx,*.EDBBlb" (without surrounding double quotes). If you have customized the table file extensions for your ElevateDB databases, then please specify the custom extensions here instead.
FileUsage	STRING This value should always be set to "1" (without surrounding double quotes).
SQLLevel	STRING This value should always be set to "0" (without surrounding double quotes).
Setup	STRING This value should always be set to the location of the ODBC driver DLL (edbodbc.dll, by default). This location can be anywhere on a local machine drive.
UsageCount	DWORD This value should always be set to "1" (without the surrounding double quotes).

Chapter 2
Using the .NET Data Provider

2.1 Application Compatibility

Supported Applications

The ElevateDB .NET Data Provider is a .NET 2.0 data provider. We have tested the data provider successfully with Microsoft Visual Studio 2005 and above.

Calling Dispose

Since the ElevateDB .NET Data Provider is indirectly accessing and using unmanaged resources during operation, you should always call the Dispose method for any EDBConnection, EDBCommand, EDBCommandBuilder, or EDBDataAdapter objects when you are done using them (deterministic destruction). Failure to do so can cause major failures in the data provider due to the resources being freed up re-entrantly when the .NET garbage collector thread finalizes these objects.

Database-Agnostic Access

The ElevateDB .NET Data Provider includes complete support for database-agnostic access via the factory class architecture in .NET 2.0, and the data provider is automatically registered as a standard data provider in the .NET 2.0 machine.config file during installation. Please see this link for more information on using the ADO.NET 2.0 factory classes:

Writing Generic Data Access Code in ASP.NET 2.0 and ADO.NET 2.0

Visual Studio Query Designer Joins

The built-in query designer in Visual Studio defaults to using SQL-89 join syntax (WHERE clause) for INNER JOINs. This is not optimal for ElevateDB because ElevateDB only optimizes joins that are specified via the SQL-92 and higher syntax of JOIN or INNER JOIN. If using INNER JOINs with your queries, please make sure to modify the SQL SELECT statement to use the INNER JOIN syntax instead in order to ensure the fastest possible query execution time. Please see the Optimizer topic for more information on the optimization of joins.

2.2 Installation and Distribution

Installation

The ElevateDB .NET Data Provider consists of one assembly called "Elevate.ElevateDB.Data.dll". By default, this assembly is automatically configured in the global .NET 2.0 machine.config file during installation so that it's codebase can be located. The data provider is not installed into the GAC (Global Assembly Cache) by default. However, the data provider is signed with a strong name key and can be installed into the GAC, if one so desires.

Distribution

The ElevateDB .NET Data Provider can be distributed royalty-free. If you wish to replicate the default installation process when distributing the data provider with your applications, you can do so by using the asblinst.exe and asblunins.exe utilities provided with the installation. The asblinst.exe utility will register the data provider as a .NET 2.0 data provider and configure the data provider's assembly codebase in the machine.config file. The asblunins.exe utility simply undoes the work done by the asblinst.exe utility. Both of these utilities can be found in the base installation directory for the product purchased. For example, if you purchased the ElevateDB DAC Standard product, then the default installation directory would be:

```
C:\Program Files\ElevateDB <Major Version> DAC-STD
```

Where <Major Version> is the major version number of the product, such as "2".

Both the asblinst.exe and the asblunins.exe utilities take the same parameters, and they are as follows (in order):

Parameter	Description
Invariant Name	This parameter specifies the invariant assembly name, and must be specified as "Elevate.ElevateDB.Data".
Description	This parameter specifies the assembly description, and should usually be specified as "ElevateDB 2 .Net Data Provider". However, it can be changed to something else if so desired.
Version	This parameter specifies the assembly version, and must be set to same value as the version number being used. The format used for version numbers in .NET assemblies is: MajorVersion.MinorVersion.BuildNumber.ReleaseNumber For example, with ElevateDB 2.03 Build 13 you would specify the version number as "2.3.13.0".
Public Key Token	This parameter specifies the public key token for the assembly, and must be specified as "cf9bc1202c75e9e2".
Codebase	This parameter specifies the location of the data provider assembly .dll file, and can be any valid path combined with the name of the data provider assembly .dll file, which is "Elevate.ElevateDB.Data.dll".

Windows Directory	This parameter specifies the system path for the Windows directory which is, by default, usually "C:\Windows" in a normal 32-bit Windows installation.
Provider Name	This parameter specifies the descriptive name of the data provider, and should usually be specified as "ElevateDB Data Provider". However, it can be changed to something else if so desired.
Factory Name	This parameter specifies the fully-qualified name of the .NET 2.0 factory class in the data provider, and must be specified as "Elevate.ElevateDB.Data.EDBProviderFactory".

For example, the default installation of the data provider would use the following command-line text to call the asblinst.exe utility:

```
asblinst.exe
"Elevate.ElevateDB.Data"
"ElevateDB 2 .Net Data Provider"
"2.3.13.0"
"cf9bc1202c75e9e2"
"C:\Program Files\ElevateDB 2
      DAC-STD\assemblies\edbprovider\Elevate.ElevateDB.Data.dll"
"C:\Windows"
"ElevateDB Data Provider"
"Elevate.ElevateDB.Data.EDBProviderFactory"
```

Note
The line breaks inserted above are only for readability and should not be used in an actual call to the asblinst utility.

2.3 Connection Strings

Connection strings are used in the EDBConnection Class component to specify information about the connection. In addition, the strongly-typed EDBConnectionStringBuilder Class component can be used to create a connection string in a safe and strongly-typed manner.

Specifying a Connection String

Connection strings specify all of the keywords necessary to configure and access a given data source via the EDBConnection component. The keywords that can be used with connection strings and the ElevateDB .NET Data Provider are listed below. Here is an example connection string that connects to a remote ElevateDB Server and a database called "Accounting" (case-insensitive):

```
CHARSET=UNICODE;
TYPE=REMOTE;
ADDRESS=192.168.0.28;
DATABASE=Accounting
```

Note
The line breaks inserted above are only for readability and should not be used in an actual connection string.

Connection String Keywords

The following keywords are used with connection strings. All keywords marked with the (R) symbol next to their name are only applicable when the TYPE keyword is set to "REMOTE". Likewise, all keywords marked with the (L) symbol next to their name are only applicable when the TYPE keyword is set to "LOCAL".

Name	Description
CHARSET	This string value specifies which character set, "ANSI" or "UNICODE", to use for the connection. For remote connections to an ElevateDB Server, the value must match the character set being used by the ElevateDB Server. The default value is "UNICODE".
TYPE	This string value is set to either "LOCAL" if the connection is accessing the database directly, or "REMOTE" if the connection is accessing the database remotely via an ElevateDB Server.
NAME	This string value specifies the name of the connection.
DESCRIPTION	This string value specifies the description of the connection.
HOST (R)	This string value specifies the host name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".

ADDRESS (R)	This string value specifies the IP address of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "127.0.0.1".
SERVICE (R)	This string value specifies the service name of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "".
PORT (R)	This string value specifies the port number of the ElevateDB Server machine that you are accessing. Either the HOST or ADDRESS values must be populated along with the SERVICE or PORT values in order to correctly access a database on an ElevateDB Server. The default value is "12010".
CONNECTTIMEOUT (R)	This string value specifies the maximum amount of time, in seconds, that ElevateDB will wait for a successful connection before aborting the connection attempt. The default value is "15" (seconds).
PING (R)	This string value specifies whether pinging will be enabled for the connection to the ElevateDB Server. When pinging is enabled, the data provider will send a keep-alive request to the ElevateDB Server in the interval specified by the PINGINTERVAL value. This prevents the ElevateDB Server from disconnecting and/or removing the connection, even if the connection has been idle for a very long period of time. Please see the Server Session Timeout configuration item in the Starting and Configuring the ElevateDB Server topic for more information on how idle connections are handled. Specify "TRUE" to enable pinging, or "FALSE" (the default) to disable pinging.
PINGINTERVAL (R)	This string value specifies how often the connection will ping the ElevateDB Server when pinging is enabled via the PING value. The default value is "60" (seconds).
TIMEOUT (R)	This string value specifies how long the connection will wait on a response from the ElevateDB Server before disconnecting and issuing an error. The default value is "180" (seconds).
ENCRYPTED (R)	This string value specifies whether the connection to the ElevateDB Server should be encrypted or no. Specify "TRUE" to enable encryption for the connection, or "FALSE" (the default) to disable encryption for the connection.
ENCRYPTSRVPWD (R)	This string value specifies the password to use for encrypted connections to the remote ElevateDB Server, as well as for encrypting logins to the remote ElevateDB Server. This password must match the configured encryption password for the remote ElevateDB Server, and you should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".
COMPRESSION (R)	This string value specifies the amount of compression to use

	when communicating with the ElevateDB Server. The default value is "0", or no compression. A value of "1" to "10" specifies the amount of compression from fast, but not very thorough, to very thorough, but not as fast. A value of "6" specifies a balance between size and speed, and represents the ideal level for most applications. **Note** The ElevateDB .NET Data Provider will automatically adjust this value for situations where the existing compression level is not ideal, such as in cases where the amount of data being sent is so small that it is of no benefit to compress the data.
TRACE (R)	This string value specifies whether tracing will be enabled for the connection to the ElevateDB Server. When tracing is enabled, the data provider will fire the EDBTrace event for every request and response to and from the ElevateDB Server. This allows the developer to examine and/or log the requests and responses in order to assist with debugging performance issues, especially over WAN connections such as the Internet. Specify "TRUE" to enable tracing, or "FALSE" (the default) to disable tracing.
CONFIGMEMORY (L)	This string value specifies whether the configuration file used by the connection will be located in the process memory ("virtual") or on disk. Specify "TRUE" to use a virtual configuration file, or "FALSE" (the default) to use a disk-based configuration file in the location specified by the CONFIGPATH string value (see below). The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on configuration files.
CONFIGPATH (L)	This string value specifies the configuration path to use for the connection. The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on the configuration path.
TEMPPATH (L)	This string value specifies the temporary tables path to use for the connection. The temporary tables path is used to store any temporary tables generated during query execution. The default value is the operating system setting for the storing temporary files for the current user. Please see the Architecture for more information on the temporary tables path.
KEEPTABLESOPEN	This string value specifies whether tables should be kept open for the duration of the connection once they have been opened at least once. Specify "TRUE" to keep tables open, or "FALSE" (the default) to have tables opened and closed on demand. Setting this value to "TRUE" can result in improved performance, especially with applications that execute many singleton SQL statements in a row.

CONFIGNAME (L)	This string value specifies the root name (without extension) used by the connection for the configuration file. The default value is "EDBConfig". The extension used for the configuration file is determined by the CONFIGEXT value. The location of the configuration file is determined by the CONFIGPATH value.
CONFIGEXT (L)	This string value specifies the extension used by the connection for the configuration file. The default value is ".EDBCfg". The root name (without extension) used for the configuration file is determined by the CONFIGNAME value. The location of the configuration file is determined by the CONFIGPATH value.
LOCKEXT (L)	This string value specifies the extension used by the connection for both the configuration and database catalog lock files. The default value is ".EDBLck". The root name (without extension) used for the configuration lock file is determined by the CONFIGNAME value. The root name (without extension) used for a database catalog lock file is determined by the CATALOGNAME value. The location of the configuration lock file is determined by the CONFIGPATH value, and the configuration lock file is hidden, by default. The location of a database catalog lock file is determined by the path designated for the applicable database when the database was created, and a database catalog lock file is hidden, by default.
LOGEXT (L)	This string value specifies the extension used by the connection for the configuration log file. The default value is ".EDBLog". The root name (without extension) used for the configuration log file is determined by the CONFIGNAME value. The location of the configuration log file is determined by the CONFIGPATH value.
MAXLOGSIZE (L)	This string value specifies the maximum size of the log file (in bytes) that the log file can grow to. Log entries are added to the log in a circular fashion, meaning that once the maximum log file size ia reached, ElevateDB will start re-using the oldest log entries for new log entries. The default value is 1048576 bytes. Which types of logged events are recorded in the log can be controlled by the LOGCATS value. By default, all categories of events are logged (INFO, WARN, and ERROR). **Warning** It is very important that all connections and/or applications accessing the same configuration file use the same maximum log file size for the configuration log file. Using different values can result in log entries being prematurely overwritten or appearing "out-of-order" when viewing the log entries via the LogEvents Table.

LOGCATS (L)	This string value specifies the types of events that should be logged in the configuration log file for the current connection, with each value separated by a comma (,). The default value is "INFO,WARN,ERROR", or all categories of events.
CATALOGNAME (L)	This string value specifies the root name (without extension) used by the connection for all database catalog files. The default value is "EDBDatabase". The extension used for the catalog files is determined by the CATALOGEXT value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
CATALOGEXT (L)	This string value specifies the extension used by the connection for database catalog files. The default value is ".EDBCat". The root name (without extension) used for all database catalog files is determined by the CATALOGNAME value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
BACKUPEXT (L)	This string value specifies the extension used by the connection for database backup files. The default value is ".EDBBkp". The root name (without extension) used for a database backup file is determined by the name given when the BACKUP DATABASE statement is executed.
UPDATEEXT (L)	This string value specifies the extension used by the connection for database update files. The default value is ".EDBUpd". The root name (without extension) used for a database update file is determined by the name given when the SAVE UPDATES statement is executed.
TABLEEXT (L)	This string value specifies the extension used by the connection for database table files. The default value is ".EDBTbl". The root name (without extension) used for a table file is determined by the name given when the CREATE TABLE statement is executed. The location of the table files is determined by the path designated for the applicable database when the database was created.
INDEXEXT (L)	This string value specifies the extension used by the connection for database table index files. The default value is ".EDBIdx". The root name (without extension) used for a table's index file is determined by the name given when the CREATE TABLE statement is executed. The location of the table index files is determined by the path designated for the applicable database when the database was created.
BLOBEXT (L)	This string value specifies the extension used by the connection for database table BLOB files. The default value is ".EDBBlb". The root name (without extension) used for a table's BLOB file is determined by the name given when the CREATE TABLE statement is executed. The location of the table BLOB files is determined by the path designated for the applicable database when the database was created.
PUBLISHEXT (L)	This string value specifies the extension used by the connection for database table publish files. The default value

	is ".EDBPbl". The root name (without extension) used for a table's publish file is determined by the name given when the CREATE TABLE statement is executed. The location of the table publish files is determined by the path designated for the applicable database when the database was created.
UID	This string value specifies the user ID to use when connecting to the connection. If this value is left blank, the user will be prompted for the user ID, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
PWD	This string value specifies the password to use when connecting to the connection. If this value is left blank, the user will be prompted for the password, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
DATABASE	This string value specifies the name of the database being accessed with the connection. Please see the Architecture for more information on databases.
READONLY	This string value specifies whether the connection is read-only or read-write. Set this value to "TRUE" to make the connection read-only, and "FALSE" to make the connection read-write.
ROWLOCKPROTOCOL	This string value specifies which row locking protocol to use. Set this value to "PESSIMISTIC" to specify that all UPDATE row locks should be acquired when the rows are read during the process of the UPDATEs. Set this value to "OPTIMISTIC" to specify that all UPDATE row locks should only be acquired when the rows are actually being updated. Please see the Locking and Concurrency topic for more information.
ROWLOCKRETRIES	This string value specifies the number of row lock attempts that the data provider should make before issuing an error. The default is "15".
ROWLOCKWAIT	This string value specifies the amount of time (in milliseconds) to wait between each row lock attempt. The default is "100".
DETECTROWCHANGES	This string value specifies whether the data provider should issue an error when a row is updated and the row has changed since it was last cached. Specify "TRUE" to enable row change detection, or "FALSE" (the default) to disable row change detection. Please see the Change Detection topic for more information.
CATALOGINFO (L)	This string value specifies whether database catalog character set and version information should appear in the Databases system information table. The default value is "TRUE".

	Note Setting the value to "FALSE" can significantly improve the performance of the loading of the Databases system information table when there are a lot of databases in a configuration. This is because ElevateDB has to open the database catalog for each database in order to read the character set and version number.
CACHEMODULES (L)	This string value specifies whether module DLLs should be cached in memory for the duration of the connection. The default value is "FALSE".
STMTCACHESIZE	This string value specifies how many SQL statements can be cached in memory for the duration of the connection. Caching SQL statements improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that SQL statements will not be cached for the connection. **Note** The maximum number of open SQL statements per connection is 2048, so you should not set the statement cache size that high. Also, the SQL statement cache size is a **per-open-database** setting.
PROCCACHESIZE	This string value specifies how many functions/procedures can be cached in memory for the duration of the connection. Caching functions/procedures improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is "0", which means that functions/procedures will not be cached for the connection. **Note** The maximum number of open functions/procedures per connection is 2048, so you should not set the procedure cache size that high. Also, the function/procedure cache size is a **per-open-database** setting.
FLUSHWRITES	This string value controls whether the data provider forces the operating system to flush any buffered writes to disk immediately after the data is written to the operating system. If the value is "FALSE" (the default), then ElevateDB will leave the flushing up to the operating system. If it is "TRUE", then ElevateDB will force a buffer flush after every write. Please see the Buffering and Caching topic for more information.
SIGNATURE	This string value specifies the signature to use for the connection. A signature is used to "sign" all configuration and database files created by ElevateDB so that they are only accessible using that signature, as well as "signing" all communications with a remote ElevateDB Server. You should

	only specify this value if you know exactly what you are doing and need to use a different signature than the default of "edb_signature".
ENCRYPTPWD (L)	This string value specifies the password to use for encrypting any local engine files. You should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".

This page intentionally left blank

Chapter 3
.NET Data Provider Reference

3.1 Introduction

The following is a detailed reference for all of the types and classes that make up the ElevateDB .NET Data Provider implementation. In the case of the common Db* classes that make up the common .NET data provider framework classes, only those members of the classes that are extensions to the basic pre-defined members are documented here. For example, because the ConnectionString property is a common property for all descendants of the DbConnection class, it is not documented here for the EDBConnection descendant class. In any case like this, please refer to the .NET 2.0 Framework documentation for information on these members.

3.2 EDBException Class

The EDBException class is used to create an instance of an exception object whenever an ElevateDB error occurs. You will find a list of all of the ElevateDB error codes in the Appendix A - Error Codes and Messages topic.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbException

Constructor

(Msg: String; Inner: Exception)

Properties

Property	Description
ErrorMsg: String	Indicates the error message that gives further information on the exception. **Note** This property is always set for every exception.
ErrorLine: Int32	Indicates the line of text in that the current exception applies to. **Note** This property may or may not be set depending upon the exception being raised.
ErrorColumn: Int32	Indicates the column of text in that the current exception applies to. **Note** This property may or may not be set depending upon the exception being raised.

3.3 EDBProviderFactory Class

The EDBProviderFactory class implements the DbProviderFactory abstract class, providing methods for creating connection, command, parameter, data adapter, command builder, and connection string builder objects.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbProviderFactory

Methods

Method	Description
CreateConnection: DbConnection	Creates a new EDBConnection instance.
CreateCommand: DbCommand	Creates a new EDBCommand instance.
CreateParameter: DbParameter	Creates a new EDBParameter instance.
CreateDataAdapter: DbDataAdapter	Creates a new EDBDataAdapter instance.
CreateCommandBuilder: DbCommandBuilder	Creates a new EDBCommandBuilder instance.
ConnectionStringBuilder: DbConnectionStringBuilder	Creates a new EDBConnectionStringBuilder instance.

3.4 EDBConnectionStringBuilder Class

The EDBConnectionStringBuilder class implements the DbConnectionStringBuilder class and provides a strongly-typed object for building an ElevateDB connection string.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbConnectionStringBuilder

Constructor

(<No Parameters>)

Properties

Property	Description
Name: String	This value specifies the name of the connection.
Description: String	This value specifies the description of the connection.
CharSet	This string value specifies which character set, "ANSI" or "Unicode", to use for the connection. For remote connections to an ElevateDB Server, the value must match the character set being used by the ElevateDB Server. This property defaults to "Unicode".
Type: String	This value is set to either "Local" if the connection is accessing the database directly, or "Remote" if the connection is accessing the database remotely via an ElevateDB Server.
ConfigMemory: Boolean	This value specifies whether the configuration file used by the connection will be located in the process memory ("virtual") or on disk. Specify True to use a virtual configuration file, or False (the default) to use a disk-based configuration file in the location specified by the ConfigPath property value (see below). The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on configuration files.
ConfigPath: String	This value specifies the configuration path to use for the connection. The configuration file in ElevateDB stores the contents of the system-defined Configuration Database. Please see the Architecture for more information on the configuration path.
TempPath: String	This value specifies the temporary tables path to use for the connection. The temporary tables path is used to store any temporary tables generated during query execution. The default value is the operating system setting for the storing temporary files for the current user. Please see the Architecture for more information on the temporary tables path.
KeepTablesOpen: Boolean	This value specifies whether tables should be kept open for the duration of the connection once they have been opened at

	least once. Specify True to keep tables open, or False (the default) to have tables opened and closed on demand. Setting this value to True can result in improved performance, especially with applications that execute many singleton SQL statements in a row.
ConfigName: String	This value specifies the root name (without extension) used by the connection for the configuration file. The default value is "EDBConfig". The extension used for the configuration file is determined by the ConfigExt property value. The location of the configuration file is determined by the ConfigPath property value.
ConfigExt: String	This value specifies the extension used by the connection for the configuration file. The default value is ".EDBCfg". The root name (without extension) used for the configuration file is determined by the ConfigName property value. The location of the configuration file is determined by the ConfigPath property value.
LockExt: String	This value specifies the extension used by the connection for both the configuration and database catalog lock files. The default value is ".EDBLck". The root name (without extension) used for the configuration lock file is determined by the ConfigName property value. The root name (without extension) used for a database catalog lock file is determined by the CatalogName property value. The location of the configuration lock file is determined by the ConfigPath property value, and the configuration lock file is hidden, by default. The location of a database catalog lock file is determined by the path designated for the applicable database when the database was created, and a database catalog lock file is hidden, by default.
LogExt: String	This value specifies the extension used by the connection for the configuration log file. The default value is ".EDBLog". The root name (without extension) used for the configuration log file is determined by the ConfigName property value. The location of the configuration log file is determined by the ConfigPath property value.
MaxLogSize: Int32	This value specifies the maximum size of the log file (in bytes) that the log file can grow to. Log entries are added to the log in a circular fashion, meaning that once the maximum log file size ia reached, ElevateDB will start re-using the oldest log entries for new log entries. The default value is 1048576 bytes. Which types of logged events are recorded in the log can be controlled by the LogCats property value. By default, all categories of events are logged (Info, Warn, and Error).

	Warning It is very important that all connections and/or applications accessing the same configuration file use the same maximum log file size for the configuration log file. Using different values can result in log entries being prematurely overwritten or appearing "out-of-order" when viewing the log entries via the LogEvents Table.
LogCats: String	This value specifies the types of events that should be logged in the configuration log file for the current connection, with each value separated by a comma (,). The default value is "Info,Warn,Error", or all categories of events.
CatalogName: String	This string value specifies the root name (without extension) used by the connection for all database catalog files. The default value is "EDBDatabase". The extension used for the catalog files is determined by the CatalogExt property value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
CatalogExt: String	This value specifies the extension used by the connection for database catalog files. The default value is ".EDBCat". The root name (without extension) used for all database catalog files is determined by the CatalogName property value. The location of the catalog file is determined by the path designated for the applicable database when the database was created.
BackupExt: String	This value specifies the extension used by the connection for database backup files. The default value is ".EDBBkp". The root name (without extension) used for a database backup file is determined by the name given when the BACKUP DATABASE statement is executed.
UpdateExt: String	This value specifies the extension used by the connection for database update files. The default value is ".EDBUpd". The root name (without extension) used for a database update file is determined by the name given when the SAVE UPDATES statement is executed.
TableExt: String	This value specifies the extension used by the connection for database table files. The default value is ".EDBTbl". The root name (without extension) used for a table file is determined by the name given when the CREATE TABLE statement is executed. The location of the table files is determined by the path designated for the applicable database when the database was created.
IndexExt: String	This value specifies the extension used by the connection for database table index files. The default value is ".EDBIdx". The root name (without extension) used for a table's index file is determined by the name given when the CREATE TABLE statement is executed. The location of the table index files is determined by the path designated for the applicable database when the database was created.

BlobExt: String	This string value specifies the extension used by the connection for database table BLOB files. The default value is ".EDBBlb". The root name (without extension) used for a table's BLOB file is determined by the name given when the CREATE TABLE statement is executed. The location of the table BLOB files is determined by the path designated for the applicable database when the database was created.
PublishExt: String	This string value specifies the extension used by the connection for database table publish files. The default value is ".EDBPbl". The root name (without extension) used for a table's publish file is determined by the name given when the CREATE TABLE statement is executed. The location of the table publish files is determined by the path designated for the applicable database when the database was created.
UID: String	This value specifies the user ID to use when connecting to the connection. If this value is left blank, the user will be prompted for the user ID, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
PWD: String	This string value specifies the password to use when connecting to the connection. If this value is left blank, the user will be prompted for the password, if possible. When executing in environments that don't support a user interface, such as web applications, it is not possible to display a login dialog and an error will be raised instead.
Database: String	This string value specifies the name of the database being accessed with the connection. Please see the Architecture for more information on databases.
ReadOnly: Boolean	This value specifies whether the connection is read-only or read-write. Set this value to True to make the connection read-only, and False to make the connection read-write.
RowLockProtocol: String	This value specifies which row locking protocol to use. Set this value to "Pessimistic" to specify that all UPDATE row locks should be acquired when the rows are read during the process of the UPDATEs. Set this value to "Optimistic" to specify that all UPDATE row locks should only be acquired when the rows are actually being updated. Please see the Locking and Concurrency topic for more information.
RowLockRetries: Int32	This value specifies the number of row lock attempts that the data provider should make before issuing an error. The default is 15.
RowLockWait: Int32	This value specifies the amount of time (in milliseconds) to wait between each row lock attempt. The default is 100.
DetectRowChanges: Boolean	This value specifies whether the data provider should issue an error when a row is updated and the row has changed since it was last cached. Specify True to enable row change detection, or False (the default) to disable row change detection. Please see the Change Detection topic for more information.
CatalogInfo: Boolean	This value specifies whether database catalog character set

	and version information should appear in the Databases system information table. The default value of this property is True. **Note** Setting this property to False can significantly improve the performance of the loading of the Databases system information table when there are a lot of databases in a configuration. This is because ElevateDB has to open the database catalog for each database in order to read the character set and version number.
CacheModules: Boolean	This string value specifies whether module DLLs should be cached in memory for the duration of the connection. The default value is False.
StmtCacheSize: Integer	This value specifies how many SQL statements can be cached in memory for the duration of the connection. Caching SQL statements improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is 0, which means that SQL statements will not be cached for the connection. **Note** The maximum number of open SQL statements per connection is 2048, so you should not set the statement cache size that high. Also, the SQL statement cache size is a **per-open-database** setting.
ProcCacheSize: Integer	This value specifies how many functions/procedures can be cached in memory for the duration of the connection. Caching functions/procedures improves the performance of ElevateDB by avoiding very expensive preparation/un-preparation cycles. The default value is 0, which means that functions/procedures will not be cached for the connection. **Note** The maximum number of open functions/procedures per connection is 2048, so you should not set the procedure cache size that high. Also, the function/procedure cache size is a **per-open-database** setting.
FlushWrites: Boolean	This value controls whether the data provider forces the operating system to flush any buffered writes to disk immediately after the data is written to the operating system. If the value is False (the default), then ElevateDB will leave the flushing up to the operating system. If it is True, then ElevateDB will force a buffer flush after every write. Please see the Buffering and Caching topic for more information.

Host: String	This value specifies the host name of the ElevateDB Server machine that you are accessing. Either the Host or Address property values must be populated along with the Service or Port property values in order to correctly access a database on an ElevateDB Server. The default value is "".
Address: String	This value specifies the IP address of the ElevateDB Server machine that you are accessing. Either the Host or Address property values must be populated along with the Service or Port property values in order to correctly access a database on an ElevateDB Server. The default value is "127.0.0.1".
Service: String	This value specifies the service name of the ElevateDB Server machine that you are accessing. Either the Host or Address property values must be populated along with the Service or Port property values in order to correctly access a database on an ElevateDB Server. The default value is "".
Port: Int32	This value specifies the port number of the ElevateDB Server machine that you are accessing Either the Host or Address property values must be populated along with the Service or Port property values in order to correctly access a database on an ElevateDB Server. The default value is 12010.
ConnectionTimeout: Int32	This value specifies the maximum amount of time, in seconds, that ElevateDB will wait for a successful connection before aborting the connection attempt. The default value is 15 (seconds).
Ping: Boolean	This value specifies whether pinging will be enabled for the connection to the ElevateDB Server. When pinging is enabled, the data provider will send a keep-alive request to the ElevateDB Server in the interval specified by the PingInterval property value. This prevents the ElevateDB Server from disconnecting and/or removing the connection, even if the connection has been idle for a very long period of time. Please see the Server Session Timeout configuration item in the Starting and Configuring the ElevateDB Server topic for more information on how idle connections are handled. Specify True to enable pinging, or False (the default) to disable pinging.
PingInterval: Int32	This value specifies how often the connection will ping the ElevateDB Server when pinging is enabled via the Ping property value. The default value is 60 (seconds).
Timeout: Int32	This value specifies how long the connection will wait on a response from the ElevateDB Server before disconnecting and issuing an error. The default value is 180 (seconds).
Encrypted: Boolean	This value specifies whether the connection to the ElevateDB Server should be encrypted or no. Specify True to enable encryption for the connection, or False (the default) to disable encryption for the connection.
EncryptSrvPwd: String	This value specifies the password to use for encrypted connections to the remote ElevateDB Server, as well as for encrypting logins to the remote ElevateDB Server. This password must match the configured encryption password for the remote ElevateDB Server, and you should only specify this value if you know exactly what you are doing and need to use

	a different encryption password than the default of "elevatesoft".
Compression: Int32	This value specifies the amount of compression to use when communicating with the ElevateDB Server. The default value is 0, or no compression. A value of 1 to 10 specifies the amount of compression from fast, but not very thorough, to very thorough, but not as fast. A value of 6 specifies a balance between size and speed, and represents the ideal level for most applications. **Note** The ElevateDB .NET Data Provider will automatically adjust this value for situations where the existing compression level is not ideal, such as in cases where the amount of data being sent is so small that it is of no benefit to compress the data.
Signature: String	This value specifies the signature to use for the connection. A signature is used to "sign" all configuration and database files created by ElevateDB so that they are only accessible using that signature, as well as "signing" all communications with a remote ElevateDB Server. You should only specify this value if you know exactly what you are doing and need to use a different signature than the default of "edb_signature".
EncryptPwd: String	This value specifies the password to use for encrypting any local engine files. You should only specify this value if you know exactly what you are doing and need to use a different encryption password than the default of "elevatesoft".

3.5 EDBLoginEventArgs Class

The EDBLoginEventArgs class is used to instantiate the parameters for the EDBLoginEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Properties

Property	Description
UserName: String	Indicates the user name to be used for the login.
Password: String	Indicates the password to be used for the login.
Continue: Boolean	Indicates whether the login should continue or not (default True).

3.6 EDBLoginEvent Delegate

The EDBLoginEvent delegate is used with the EDBConnection OnLogin event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBLoginEventArgs)

3.7 EDBTimeoutEventArgs Class

The EDBTimeoutEventArgs class is used to instantiate the parameters for the EDBTimeoutEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Properties

Property	Description
StayConnected: Boolean	Indicates whether the connection should keep waiting for a response from the ElevateDB Server by staying connected, or disconnect from the ElevateDB Server (default True).

3.8 EDBTimeoutEvent Delegate

The EDBTimeoutEvent delegate is used with the EDBConnection OnTimeout event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBTimeoutEventArgs)

3.9 EDBReconnectEventArgs Class

The EDBReconnectEventArgs class is used to instantiate the parameters for the EDBReconnectEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Properties

Property	Description
Continue: Boolean	Indicates whether the reconnection should continue or not (default True).
StopAsking: Boolean	Indicates whether the OnReconnect event should stop being triggered until the connnection is subsequently disconnected (default False).

3.10 EDBReconnectEvent Delegate

The EDBReconnectEvent delegate is used with the EDBConnection OnReconnect event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBReconnectEventArgs)

3.11 EDBCommsProgressEventArgs Class

The EDBCommsProgressEventArgs class is used to instantiate the parameters for the EDBCommsProgressEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Properties

Property	Description
NumBytes: Int32	Indicates the number of bytes sent/received so far for the current request/response to/from the ElevateDB Server.
PercentDone: Int32	Indicates the percentage completed for the current request/response.

3.12 EDBCommsProgressEvent Delegate

The EDBCommsProgressEvent delegate is used with the EDBConnection OnSendProgress and OnReceiveProgress events.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBCommsProgressEventArgs)

3.13 EDBTraceEventArgs Class

The EDBTraceEventArgs class is used to instantiate the parameters for the EDBTraceEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Properties

Property	Description
DateTime: DateTime	Indicates the date and time of the request/response.
ElapsedTime: UInt32	Indicates the total elapsed time in milliseconds for the request/response.
Compression: Int32	Indicates the current compression level for the request/response. This value normally ranges from 0 (no compression) to 9 (best compression), but in some cases may actually appear in the trace record as values greater than or equal to 10. In these cases, the compression has been adjusted by the engine due to the size of the data being too small (less than 1024 bytes). The adjusted compression level can be found by doing this calculation: Compression mod 10 And the original compression level before the adjustment can be found by using the following calculation: Compression div 10 Any adjustments to the compression such as this are active for the current request/response only and do not persist any further.
FunctionCode: Int32	Indicates the function ID of the request.
FunctionName: String	Indicates the function name of the request.
ResultCode: Int32	Indicates the result code of the request/response. If this value is -1, then the trace record represents a request. Any 0 or higher value indicates that the trace record is a response. You can use this field to determine whether the trace record is for a request (-1) or a response (0 or higher).
Size: Int32	Indicates the request/response size, in bytes.

3.14 EDBTraceEvent Delegate

The EDBTraceEvent delegate is used with the EDBConnection OnTrace event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBTraceEventArgs)

3.15 EDBProgressEventArgs Class

The EDBProgressEventArgs class is used to instantiate the parameters for the EDBProgressEvent delegate.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Constructors

(ProgressPercentDone: Int32)

Properties

Property	Description
PercentDone: Int32	Indicates the percentage completed for the current command execution.
Continue: Boolean	Indicates whether the command execution should continue or not.

3.16 EDBProgressEvent Delegate

The EDBProgressEvent delegate is used with the EDBCommand OnProgress event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBProgressEventArgs)

3.17 EDBMessageEventArgs Class

The EDBMessageEventArgs class is used to instantiate the parameters for the EDBStatusEvent and EDBLogEvent delegates.

Namespace: Elevate.ElevateDB.Data

Inherits From System.EventArgs

Constructors

(Msg: String)

Properties

Property	Description
Message: String	Indicates the current status message being returned for the command execution.

3.18 EDBStatusEvent Delegate

The EDBStatusEvent delegate is used with the EDBCommand OnStatusMessage event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBMessageEventArgs)

3.19 EDBLogEvent Delegate

The EDBLogEvent delegate is used with the EDBCommand OnLogMessage event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object; EventArgs: EDBMessageEventArgs)

3.20 EDBType Enumeration

The EDBType enumeration is used to indicate the native ElevateDB data type of parameters and/or columns. For more information on the types available in ElevateDB, please see the Types topic.

Namespace: Elevate.ElevateDB.Data

Member	Description
Unknown	Indicates an unknown type.
Char	Indicates a CHAR type.
VarChar	Indicates a VARCHAR type.
Guid	Indicates a GUID type.
Byte	Indicates a BYTE type.
VarByte	Indicates a VARBYTE type.
Blob	Indicates a BLOB type.
Clob	Indicates a CLOB type.
Bool	Indicates a BOOLEAN type.
SmallInt	Indicates a SMALLINT type.
Int	Indicates an INT type.
BigInt	Indicates a LARGEINT type.
Float	Indicates a FLOAT or DOUBLE type.
Decimal	Indicates a DECIMAL type.
Date	Indicates a DATE type.
Time	Indicates a TIME type.
TimeStamp	Indicates a TIMESTAMP type.
IntervalYear	Indicates an INTERVAL YEAR type.
IntervalYearMonth	Indicates an INTERVAL YEAR TO MONTH type.
IntervalMonth	Indicates an INTERVAL MONTH type.
IntervalDay	Indicates an INTERVAL DAY type.
IntervalDayHour	Indicates an INTERVAL DAY TO HOUR type.
IntervalDayMin	Indicates an INTERVAL DAY TO MINUTE type.
IntervalDaySec	Indicates an INTERVAL DAY TO SECOND type.
IntervalDayMSec	Indicates an INTERVAL DAY TO MSECOND type.
IntervalHour	Indicates an INTERVAL HOUR type.
IntervalHourMin	Indicates an INTERVAL HOUR TO MINUTE type.
IntervalHourSec	Indicates an INTERVAL HOUR TO SECOND type.
IntervalHourMSec	Indicates an INTERVAL HOUR TO MSECOND type.

IntervalMin	Indicates an INTERVAL MINUTE type.
IntervalMinSec	Indicates an INTERVAL MINUTE TO SECOND type.
IntervalMinMSec	Indicates an INTERVAL MINUTE TO MSECOND type.
IntervalSec	Indicates an INTERVAL SECOND type.
IntervalSecMSec	Indicates an INTERVAL SECOND TO MSECOND type.
IntervalMSec	Indicates an INTERVAL MSECOND type.

3.21 EDBParameter Class

The EDBParameter class extends the DbParameter class, providing an encapsulation of an ElevateDB parameter.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbParameter

Implements ICloneable

Constructors

(<No Parameters>)
(ParameterName: String; Type: DbType)
(ParameterName: String; Value: System.Object)
(ParameterName: String; Type: DbType; SourceColumn: String)

Properties

Property	Description
ProviderType: EDBType	Indicates the native ElevateDB type of the parameter.
Precision: Byte	Indicates the precision of an approximate or exact numeric type.
Scale: Byte	Indicates the scale of an exact numeric type.
Size: Int32	Indicates the size of a binary type in bytes or the length of a character type in characters.

3.22 EDBTransaction Class

The EDBTransaction class implements the DbTransaction abstract class, providing an encapsulation of an ElevateDB transaction on a given ElevateDB database.

> **Note**
> Since ElevateDB only supports an isolation level of Serializable, the isolation level is not used as a parameter for alternative constructors such as the constructor for starting a restricted transaction. Please see the Transactions for more information.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbTransaction

Constructors

(<No Parameters>)
(Connection: EDBConnection)
(Connection: EDBConnection; IsolationLevel: IsolationLevel)
(Connection: EDBConnection; const Tables: array of String)

3.23 EDBYearMonthIntervalType Enumeration

The EDBYearMonthIntervalType enumeration is used to indicate the type of year-month intervals when using the YearMonthIntervalToSQLStr and SQLStrToYearMonthInterval methods of the EDBConnection Class class.

Namespace: Elevate.ElevateDB.Data

Member	Description
Year	Indicates a year interval.
Month	Indicates a month interval.
YearMonth	Indicates a year-month interval.

3.24 EDBDayTimeIntervalType Enumeration

The EDBDayTimeIntervalType enumeration is used to indicate the native ElevateDB data type of parameters and/or columns. For more information on the types available in ElevateDB, please see the Types topic.

Namespace: Elevate.ElevateDB.Data

Member	Description
Day	Indicates a day interval.
Hour	Indicates an hour interval.
Minute	Indicates a minute interval.
Second	Indicates a second interval.
MSecond	Indicates a millisecond interval.
DayHour	Indicates a day-hour interval.
DayMinute	Indicates a day-minute interval.
DaySecond	Indicates a day-second interval.
DayMSecond	Indicates a day-millisecond interval.
HourMinute	Indicates an hour-minute interval.
HourSecond	Indicates an hour-second interval.
HourMSecond	Indicates an hour-millisecond interval.
MinuteSecond	Indicates a minute-second interval.
MinuteMSecond	Indicates a minute-millisecond interval.
SecondMSecond	Indicates a second-millisecond interval.

3.25 EDBConnection Class

The EDBConnection class implements the DbConnection abstract class, providing an encapsulation of an ElevateDB local or remote session connected to a given ElevateDB database. A session acts like a "virtual user" and each new EDBConnection instance used in an application maintains its own database connections, table buffers, table/view/query result set cursors, etc. Because of the unique requirements of a multi-threaded application, ElevateDB requires that you use a separate EDBConnection component for each thread in use, thus treating each thread as a separate "virtual user".

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbConnection

Implements ICloneable

Constructors

(<No Parameters>)
(ConnectString: String)

Properties

Property	Description
EngineVersion: String	Indicates the current version of ElevateDB being used. This property is read-only.
InTransaction: Boolean	Use the InTransaction property at run-time to determine if a transaction is currently in progress for the current connection. The InTransaction property is True if a transaction is in progress and False if a transaction is not in progress. The value of the InTransaction property cannot be changed directly. Calling the EDBConnection BeginTransaction method sets the InTransaction property to True. Calling the EDBTransaction Commit or EDBTransaction Rollback methods sets the InTransaction property to False.
ReadOnly: Boolean	Indicates whether the current connection is using read-only access only.
ServerDateTime: DateTime	Retrieves the server date/time in local time.
ServerUTCDateTime: DateTime	Retrieves the server date/time in UTC time.
ServerName: String	Retrieves the remote server name.
ServerDescription: String	Retrieves the remote server description.
RemoteParameters: EDBParameterCollection	Use the RemoteParameters property at runtime to specify the parameters to be used with the CallRemoteProcedure when calling a custom server procedure. This property can also be used to retrieve any return values from the custom server procedure (parameter Direction property of ReturnValue).

Methods

Method	Description
Clone: Object	Clones the current connection, including current contents of the ConnectionString property.
BeginTransaction(array of String): DbTransaction	Starts a restricted transaction for the current connection. Please see the Transactions topic for more information on restricted transactions.
FreeCachedStatements(DatabaseName: String)	Frees any cached SQL statements for the specified open database. If a database is not specified, then any cached SQL statements in the open databases for the connection will be freed. Please see the Buffering and Caching topic for more information on caching SQL statements.
FreeCachedProcedures(DatabaseName: String)	Frees any cached functions/procedures for the specified open database. If a database is not specified, then any cached functions/procedures in the open databases for the connection will be freed. Please see the Buffering and Caching topic for more information on caching functions/procedures.
QuotedSQLStr(Value: String): String	Formats a string constant so that it can properly used as a literal constant in an SQL statement. This method converts escapes all single quotes and converts all characters less than #32 (space) into the #<ASCII value> syntax.
DateToSQLStr(Value: DateTime): String	Converts a DateTime value to an SQL 2003 standard date constant string. All SQL and filter expressions in ElevateDB require standard date constants which use the 'yyyy-mm-dd' format.
TimeToSQLStr(Value: DateTime; MilitaryTime: Boolean): String	Converts a DateTime value to an SQL 2003 standard time constant string. All SQL and filter expressions in ElevateDB require standard time constants which use the 'hh:mm:ss.zzz am/pm' format. Use the MilitaryTime parameter to indicate whether the time should be returned in 24-hour format instead of 12-hour format with an am/pm indicator.
DateTimeToSQLStr(Value: DateTime; MilitaryTime: Boolean): String	Converts a DateTime value to an SQL 2003 standard timestamp constant string. All SQL and filter expressions in ElevateDB require standard timestamp constants which use the 'yyyy-mm-dd hh:mm:ss.zzz am/pm' format. Use the MilitaryTime parameter to indicate whether the time should be returned in 24-hour format instead of 12-hour format with an am/pm indicator.
SQLStrToDate(Value: String): DateTime	Converts a string that contains an SQL 2003 standard date constant to an actual TDateTime value. All SQL and filter expressions in ElevateDB require standard date constants which use the 'yyyy-mm-dd' format.
SQLStrToTime(Value: String): DateTime	Converts a string that contains an SQL 2003 standard time constant to an actual TDateTime value. All SQL and filter expressions in ElevateDB require standard time constants which use the 'hh:mm:ss.zzz am/pm' format.

SQLStrToDateTime(Value: String): DateTime	Converts a string that contains an SQL 2003 standard timestamp constant to an actual TDateTime value.All SQL and filter expressions in ElevateDB require standard timestamp constants which use the 'yyyy-mm-dd hh:mm:ss.zzz am/pm' format.
YearMonthIntervalToSQLStr(Value: Int32; YearMonthIntervalType: EDBYearMonthIntervalType): String	Converts a native year-month value to an SQL 2003 standard year-month interval constant string. All SQL and filter expressions in ElevateDB require standard year-month interval constants which use the general 'yyyy-mm' format. Use the YearMonthIntervalType parameter to indicate how the year-month interval should be formatted.
SQLStrToYearMonthInterval(Value: String; YearMonthIntervalType: EDBYearMonthIntervalType): Int32	Converts a string that contains an SQL 2003 standard year-month interval constant to a native year-month value. All SQL and filter expressions in ElevateDB require standard year-month interval constants which use the 'yyyy-mm' format.
DayTimeIntervalToSQLStr(Value: Int64; DayTimeIntervalType: EDBDayTimeIntervalType): String	Converts a native day-time value to an SQL 2003 standard day-time interval constant string. All SQL and filter expressions in ElevateDB require standard day-time interval constants which use the general 'dd hh:mm:ss.zzz am/pm' format. Use the DayTimeIntervalType parameter to indicate how the day-time interval should be formatted.
SQLStrToDayTimeInterval(Value: String; DayTimeIntervalType: EDBDayTimeIntervalType): Int64	Converts a string that contains an SQL 2003 standard day-time interval constant to a native day-time value. All SQL and filter expressions in ElevateDB require standard day-time interval constants which use the 'dd hh:mm:ss.zzz am/pm' format.
BooleanToSQLStr(Value: Boolean): String	Converts a Boolean value to an SQL 2003 standard boolean constant string. All SQL and filter expressions in ElevateDB require standard boolean constants, which are TRUE and FALSE (case-insensitive).
SQLStrToBoolean(Value: String): Boolean	Converts a string that contains an SQL 2003 standard boolean constant to an actual Boolean value. All SQL and filter expressions in ElevateDB require standard boolean constants, which are TRUE and FALSE (case-insensitive).
FloatToSQLStr(Value: Double): String	Converts a Double value to an SQL 2003 standard float constant string. All SQL and filter expressions in ElevateDB require standard float constants which use the period (.) as the decimal separator.
SQLStrToFloat(Value: String): Double	Converts a string that contains an SQL 2003 standard float constant to an actual Double value. All SQL and filter expressions in ElevateDB require standard float constants which use the period (.) as the decimal separator.
DecimalToSQLStr(Value: Decimal; Scale: Integer): String	Converts a Decimal value to an SQL 2003 standard decimal constant string. All SQL and filter expressions in ElevateDB require standard decimal constants which use the period (.) as the decimal separator. Use the Scale parameter to specify the number of decimal places to use for the output string, or 0 to specify that the number of decimal places in the output string will depend upon the Decimal value being converted.

SQLStrToDecimal(Value: String): Decimal	Converts a string that contains an SQL 2003 standard decimal constant to an actual Decimal value. All SQL and filter expressions in ElevateDB require standard decimal constants which use the period (.) as the decimal separator.
BinaryToSQLStr(Value: array of Byte): String	Converts a byte array value to an SQL 2003 standard binary constant string. All SQL and filter expressions in ElevateDB require standard binary constants, which are represented by the binary value in hexadecimal format.
SQLStrToBinary(Value: String): array of Byte	Converts a string that contains an SQL 2003 standard binary constant to an actual byte array value. All SQL and filter expressions in ElevateDB require standard binary constants, which are represented by the binary value in hexadecimal format.
CallRemoteProcedure(NameOfProcedure: String)	Use the CallRemoteProcedure method along with the RemoteParameters property to call a custom server procedure in a remote ElevateDB Server. **Note** The parameters that are sent along with the custom server procedure call are completely user-defined. It is up to the calling remote session to define all necessary parameters, including output or result parameters for getting data back from the custom server procedure.
CancelRemoteProcedure	Cancels any currently-executing remote procedure. **Note** This method is simply an indication that the client application wants to cancel the remote procedure. The actual server procedure may or may not send progress messages that allow for cancellation, and it may or may not actually respond to a cancellation attempt. It is entirely up to the developer of the server procedure as to how the procedure behaves in the face of a cancellation/abort attempt.

Events

Event	Description
OnLogin: EDBLoginEvent	Setting this event adds an event handler to the list of event handlers listening for the OnLogin event when the connection is first opened. The arguments for the event handler are defined in the EDBLoginEventArgs class. The OnLogin event is used to augment the login process and programmtically supply the user name and password used with the login. It can also be used to display a custom interactive login dialog whenever a connection is first opened.

	Note The OnLogin event is only triggered when either the user name (UID keyword) or password (PWD keyword) are missing from the connection string used with the connection. Please see the Connection Strings topic for more information on the UID and PWD connection string keywords.
OnTimeout: EDBTimeoutEvent	Setting this event adds an event handler to the list of event handlers listening for the OnTimeout event while the current connection is connected to an ElevateDB Server. The arguments for the event handler are defined in the EDBTimeoutEventArgs class. The OnTimeout event is used to deal with situations where the connection is waiting on a response from an ElevateDB Server, and the wait time has exceeded the timeout (TIMEOUT keyword) defined for the connection in the connection string. Setting the StayConnected property of the event arguments to False will cause the connection to disconnect from the ElevateDB Server, whereas leaving the StayConnected property of the event arguments to True will keep the connection connected to the ElevateDB Server. Please see the Connection Strings topic for more information on the TIMEOUT connection string keyword.
OnReconnect: EDBReconnectEvent	Setting this event adds an event handler to the list of event handlers listening for the OnReconnect event while the current connection is connected to an ElevateDB Server. The arguments for the event handler are defined in the EDBReconnectEventArgs class. The OnReconnect event is called whenever the connection needs to be re-established automatically due to a drop in the connection or a timeout (see OnTimeout event above). This event usually only occurs when the connection was not configured to ping the ElevateDB Server at regular intervals in the connection string (PING keyword set to False). In such a case, the ElevateDB Server may timeout the connection and disconnect it (but not remove it), resulting in the OnReconnect event being triggered the next time the connection tries to send a request to the ElevateDB Server. Please see the Connection Strings topic for more information on the PING connection string keyword.
OnSendProgress: EDBCommsProgressEvent	Setting this event adds an event handler to the list of event handlers listening for the OnSendProgress event while the current connection is connected to an ElevateDB Server. The arguments for the event handler are defined in the EDBCommsProgressEventArgs class. The OnSendProgress event is called whenever the connection needs to send a request to an ElevateDB Server, and may be called multiple times for a single request, depending upon the size of the request.
OnReceiveProgress: EDBCommsProgressEvent	Setting this event adds an event handler to the list of event handlers listening for the OnReceiveProgress event while the current connection is connected to an ElevateDB Server. The

	arguments for the event handler are defined in the EDBCommsProgressEventArgs class. The OnReceiveProgress event is called whenever the connection receives a response to a request from an ElevateDB Server, and may be called multiple times for a single response, depending upon the size of the response.
OnTrace: EDBTraceEvent	Setting this event adds an event handler to the list of event handlers listening for the OnTrace event while the current connection is connected to an ElevateDB Server. The arguments for the event handler are defined in the EDBTraceEventArgs class. The OnTrace event is called whenever the connection sends a request to an ElevateDB Server or receives a response to a request from an ElevateDB Server. It is useful for tracing the activity for a connection to help determine performance bottlenecks, especially over WANs (wide-area networks) like the Internet.
OnRemoteProgress: EDBProgressEvent	Setting this event adds an event handler to the list of event handlers listening for the OnRemoteProgress event during the execution of a remote server procedure. The arguments for the event handler are defined in the EDBProgressEventArgs class. The OnRemoteProgress event is used to retrieve the progress of the currently-executing procedure and, optionally, to cancel the execution of the procedure. **Note** The actual server procedure may or may not send progress messages that allow for cancellation, and it may or may not actually respond to a cancellation attempt. It is entirely up to the developer of the server procedure as to how the procedure behaves in the face of a cancellation/abort attempt.
OnRemoteStatusMessage: EDBStatusMessageEvent	Setting this event adds an event handler to the list of event handlers listening for the OnRemoteStatusMessage event during the execution of a remote server procedure. The arguments for the event handler are defined in the EDBMessageEventArgs class. The OnRemoteStatus event is used to retrieve the status of the currently-executing procedure.
OnRemoteLogMessage: EDBLogMessageEvent	Setting this event adds an event handler to the list of event handlers listening for the OnLogMessage event during the execution of a remote server procedure. The arguments for the event handler are defined in the EDBMessageEventArgs class. The OnRemoteLogMessage event is used to retrieve any log messages for the currently-executing procedure.

3.26 EDBCommandTextType Enumeration

The EDBCommandTextType enumeration is used to indicate what type of command is currently specified via the CommandText property of an EDBCommand instance when the CommandType property is set to Text.

Namespace: Elevate.ElevateDB.Data

Member	Description
Empty	Indicates that the current command text is empty.
Select	Indicates that the current command text is a SELECT statement.
Insert	Indicates that the current command text is an INSERT statement.
Update	Indicates that the current command text is an UPDATE statement.
Delete	Indicates that the current command text is a DELETE statement.
CreateDatabase	Indicates that the current command text is a CREATE DATABASE statement.
CreateUser	Indicates that the current command text is a CREATE USER statement.
CreateRole	Indicates that the current command text is a CREATE ROLE statement.
CreateJob	Indicates that the current command text is a CREATE JOB statement.
CreateStore	Indicates that the current command text is a CREATE STORE statement.
CreateModule	Indicates that the current command text is a CREATE MODULE statement.
CreateTextFilter	Indicates that the current command text is a CREATE TEXT FILTER statement.
CreateWordGenerator	Indicates that the current command text is a CREATE WORD GENERATOR statement.
CreateMigrator	Indicates that the current command text is a CREATE MIGRATOR statement.
CreateTable	Indicates that the current command text is a CREATE TABLE statement.
CreateView	Indicates that the current command text is a CREATE VIEW statement.
CreateIndex	Indicates that the current command text is a CREATE INDEX statement.

CreateTrigger	Indicates that the current command text is a CREATE TRIGGER statement.
CreateTextIndex	Indicates that the current command text is a CREATE TEXT INDEX statement.
CreateFunction	Indicates that the current command text is a CREATE FUNCTION statement.
CreateProcedure	Indicates that the current command text is a CREATE PROCEDURE statement.
DropDatabase	Indicates that the current command text is a DROP DATABASE statement.
DropUser	Indicates that the current command text is a DROP USER statement.
DropRole	Indicates that the current command text is a DROP ROLE statement.
DropJob	Indicates that the current command text is a DROP JOB statement.
DropStore	Indicates that the current command text is a DROP STORE statement.
DropModule	Indicates that the current command text is a DROP MODULE statement.
DropTextFilter	Indicates that the current command text is a DROP TEXT FILTER statement.
DropWordGenerator	Indicates that the current command text is a DROP WORD GENERATOR statement.
DropMigrator	Indicates that the current command text is a DROP MIGRATOR statement.
DropTable	Indicates that the current command text is a DROP TABLE statement.
DropView	Indicates that the current command text is a DROP VIEW statement.
DropIndex	Indicates that the current command text is a DROP INDEX statement.
DropTrigger	Indicates that the current command text is a DROP TRIGGER statement.
DropFunction	Indicates that the current command text is a DROP FUNCTION statement.
DropProcedure	Indicates that the current command text is a DROP PROCEDURE statement.
AlterDatabase	Indicates that the current command text is an ALTER DATABASE statement.
AlterUser	Indicates that the current command text is an ALTER USER statement.
AlterRole	Indicates that the current command text is an ALTER ROLE statement.

AlterJob	Indicates that the current command text is an ALTER JOB statement.
AlterStore	Indicates that the current command text is an ALTER STORE statement.
AlterModule	Indicates that the current command text is an ALTER MODULE statement.
AlterTextFilter	Indicates that the current command text is an ALTER TEXT FILTER statement.
AlterWordGenerator	Indicates that the current command text is an ALTER WORD GENERATOR statement.
AlterMigrator	Indicates that the current command text is an ALTER MIGRATOR statement.
AlterTable	Indicates that the current command text is an ALTER TABLE statement.
AlterView	Indicates that the current command text is an ALTER VIEW statement.
AlterIndex	Indicates that the current command text is an ALTER INDEX statement.
AlterTrigger	Indicates that the current command text is an ALTER TRIGGER statement.
AlterFunction	Indicates that the current command text is an ALTER FUNCTION statement.
AlterProcedure	Indicates that the current command text is an ALTER PROCEDURE statement.
RenameDatabase	Indicates that the current command text is an RENAME DATABASE statement.
RenameUser	Indicates that the current command text is an RENAME USER statement.
RenameRole	Indicates that the current command text is an RENAME ROLE statement.
RenameJob	Indicates that the current command text is an RENAME JOB statement.
RenameStore	Indicates that the current command text is an RENAME STORE statement.
RenameModule	Indicates that the current command text is an RENAME MODULE statement.
RenameTextFilter	Indicates that the current command text is an RENAME TEXT FILTER statement.
RenameWordGenerator	Indicates that the current command text is an RENAME WORD GENERATOR statement.
RenameMigrator	Indicates that the current command text is an RENAME MIGRATOR statement.
RenameTable	Indicates that the current command text is an RENAME TABLE statement.

RenameView	Indicates that the current command text is an RENAME VIEW statement.
RenameIndex	Indicates that the current command text is an RENAME INDEX statement.
RenameTrigger	Indicates that the current command text is an RENAME TRIGGER statement.
RenameFunction	Indicates that the current command text is an RENAME FUNCTION statement.
RenameProcedure	Indicates that the current command text is an RENAME PROCEDURE statement.
Grant	Indicates that the current command text is a GRANT PRIVILEGES or GRANT ROLES statement.
Revoke	Indicates that the current command text is a REVOKE PRIVILEGES or REVOKE ROLES statement.
RepairTable	Indicates that the current command text is a REPAIR TABLE statement.
VerifyTable	Indicates that the current command text is a VERIFY TABLE statement.
OptimizeTable	Indicates that the current command text is an OPTIMIZE TABLE statement.
EmptyTable	Indicates that the current command text is an EMPTY TABLE statement.
ExportTable	Indicates that the current command text is an EXPORT TABLE statement.
ImportTable	Indicates that the current command text is an IMPORT TABLE statement.
SetBackupsStore	Indicates that the current command text is a SET BACKUPS STORE statement.
BackupDatabase	Indicates that the current command text is a BACKUP DATABASE statement.
RestoreDatabase	Indicates that the current command text is a RESTORE DATABASE statement.
SetUpdatesStore	Indicates that the current command text is a SET UPDATES STORE statement.
SaveDatabaseUpdates	Indicates that the current command text is a SAVE UPDATES statement.
LoadDatabaseUpdates	Indicates that the current command text is a LOAD UPDATES statement.
PublishDatabase	Indicates that the current command text is a PUBLISH DATABASE statement.
UnpublishDatabase	Indicates that the current command text is an UNPUBLISH DATABASE statement.
MigrateDatabase	Indicates that the current command text is a MIGRATE DATABASE statement.

DisconnectServerSession	Indicates that the current command text is a DISCONNECT SERVER SESSION statement.
RemoveServerSession	Indicates that the current command text is a REMOVE SERVER SESSION statement.
SetFilesStore	Indicates that the current command text is a SET FILES STORE statement.
CopyFile	Indicates that the current command text is a COPY FILE statement.
RenameFile	Indicates that the current command text is a RENAME FILE statement.
DeleteFile	Indicates that the current command text is a DELETE FILE statement.
SetInformationCollate	Indicates that the current command text is a SET INFORMATION COLLATE statement.
CompareDatabase	Indicates that the current command text is a COMPARE DATABASE statement.
Script	Indicates that the current command text is a script.

3.27 EDBCommand Class

The EDBCommand class implements the DbCommand abstract class, providing an encapsulation of an ElevateDB table, stored procedure, SQL statement, or script.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbCommand

Implements ICloneable

Constructors

(<No Parameters>)
(CommandText: String)
(CommandText: String; Connection: EDBConnection)
(CommandText: String; Connection: EDBConnection; Transaction: EDBTransaction)

Properties

Property	Description
CommandTextType: EDBCommandTextType	Indicates the type of command specified in the CommandText property when the CommandType property is set to Text.
EngineVersion: String	Indicates the current version of ElevateDB being used. This property is read-only.
ExecutionResult: Boolean	The ExecutionResult property indicates the result of the execution of the current command. Currently, this only applies to the VERIFY TABLE and REPAIR TABLE SQL statements. This property will be True if any errors were found, and False if no errors were found.
ExecutionTime: Double	Indicates the total time, in seconds, that the current command took to execute when the CommandType property is set to Text. This time does not include any time taken to prepare the command, if required, only the execution time itself.
IsPrepared: Boolean	Indicates whether the current command text has been prepared or not.
IsOpened: Boolean	Indicates whether there is an active data reader present for the current command.
ParamCheck: Boolean	Specifies whether or not the Parameters property is cleared and regenerated if the CommandText property is modified. By default the ParamCheck property is False, meaning that the Parameters property is not automatically generated. When ParamCheck is True, the proper number of parameters is guaranteed to be generated for the current command.

Plan: String	Indicates the query plan for the current SQL statement specified in the CommandText property once the statement is executed and the RequestPlan property is set to True. The Plan property is cleared before each new statement execution.
RequestPlan: Boolean	Specifies that a query plan be generated and stored in the Plan property when the SQL statement specified in the CommandText property is executed. The default is False. **Note** Query plans are only generated for SQL SELECT, INSERT, UPDATE, or DELETE statements.
RequestSensitive: Boolean	Specifies whether or not ElevateDB should attempt to return a sensitive result set when the current SELECT statement in the CommandText property is excuted. The RequestSensitive property is False by default, meaning that an insensitive and read-only result set will be returned. **Note** Setting RequestSensitive to True does not guarantee that a sensitive result set will be returned by ElevateDB. A sensitive result set will be returned only if the SELECT statement syntax conforms to the syntax requirements for a sensitive result set. If the RequestSensitive property is True, but the syntax does not conform to the requirements, ElevateDB returns an insensitive result set. After executing the query, inspect the Sensitive property to determine whether the request for a sensitive result set was successful.
RowReadSize: Int32	Use this property to specify how many rows should be read from an ElevateDB Server at one time when any result set for the command is navigated. This property is useful for specifying the row read size when you cannot get access to the EDBDataReader instance used for navigating the result set, which is the case when using an EDBDataAdapter instance. **Note** Any time an EDBDataReader or EDBDataCursor instance is created using the ExecuteReader or ExecuteCursor (ElevateDB-specific) methods, the RowReadSize property of the instance will be initialized with the value specified by the EDBCommand.RowReadSize property.
RowsAffected: Int32	Indicates how many rows were selected, inserted, updated or deleted by the execution of the current SQL statement

	specified via the CommandText property. If RowsAffected is 0, the SQL statement did not select, insert, update or delete any rows. **Note** This property is only useful for SELECT, INSERT, UPDATE, or DELETE statements.
Sensitive: Boolean	Indicates whether the current SELECT statement returned a sensitive result set.

Methods

Method	Description
ExecuteCursor: EDBDataCursor	Executes the current command, and returns an instance of the EDBDataCursor class that can be used as a bi-directional cursor on the result set generated by the command.
Close	Closes the current command, freeing any result set that may be active from the last execution of the command.
LoadFromFile(const FileName: String)	Loads a command from a text file into the CommandText property of the current command.

Events

Event	Description
OnProgress: EDBProgressEvent	Setting this event adds an event handler to the list of event handlers listening for the OnProgress event during the current command execution. The arguments for the event handler are defined in the EDBProgressEventArgs class. The OnProgress event is used to retrieve the progress of the currently-executing command and, optionally, to cancel the execution of the command.
OnStatusMessage: EDBStatusMessageEvent	Setting this event adds an event handler to the list of event handlers listening for the OnStatusMessage event during the current command execution. The arguments for the event handler are defined in the EDBMessageEventArgs class. The OnStatus event is used to retrieve the status of the currently-executing command.
OnLogMessage: EDBLogMessageEvent	Setting this event adds an event handler to the list of event handlers listening for the OnLogMessage event during the current command execution. The arguments for the event handler are defined in the EDBMessageEventArgs class. The OnLogMessage event is used to retrieve any log messages for the currently-executing command. Log messages are generated during various administrative and DDL statements.

3.28 EDBDataReader Class

The EDBDataReader class implements the DbDataReader abstract class, providing an encapsulation of a data reader object for enumerating the result set of an EDBCommand instance.

> **Note**
> In addition to the normal Get* methods that use column indexes to retrieve column data, there are also matching Get* methods that accept a column name instead of a column index. The only difference in the method parameters for these methods is the first parameter, which is a String instead of an Int32.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbDataReader

Constructors

(<No Parameters>)
(Command: EDBCommand; CommandBehavior: System.Data.CommandBehavior)

Properties

Property	Description
RowReadSize: Integer	Use this property to specify how many rows should be read from an ElevateDB Server at one time when the current result set is navigated.

Methods

Method	Description
Refresh	Refreshes the current result set. This is useful when the result set is sensitive, as indicated by the EDBCommand Sensitive property
GetBytes(Index: Int32; DestStream: Stream): Int64 GetBytes(Index: Int32; const DestFileName: String): Int64	These methods retrieve the contents of a CHAR/VARCHAR, BYTES/VARBYTES, CLOB, or BLOB column into a destination stream or file. The return value is the number of bytes read from the column.
GetImage(Index: Int32): Bitmap	This method retrieves the contents of a BLOB column into a new Bitmap instance.

3.29 EDBDataCursorState Enumeration

The EDBDataCursorState enumeration is used to indicate the state of a data cursor.

Namespace: Elevate.ElevateDB.Data

Member	Description
Browsing	Indicates that the current data cursor is browsing.
Inserting	Indicates that the current data cursor is inserting a new row.
Updating	Indicates that the current data cursor is updating an existing row.

3.30 EDBCursorStateChangeEvent Delegate

The EDBDataCursorStateChangeEvent delegate is used with the EDBDataCursor OnStateChangeEvent event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object)

3.31 EDBCursorMoveEvent Delegate

The EDBDataCursorMoveEvent delegate is used with the EDBDataCursor OnMoveEvent event.

Namespace: Elevate.ElevateDB.Data

Parameters

(Sender: System.Object)

3.32 EDBDataCursor Class

The EDBDataCursor class descends from the EDBDataReader class, providing a bi-directional cursor that can be created using the ExecuteCursor method of the EDBCommand class. The EDBDataCursor class also supports in-place inserts, updates, and deletes, provided that the result set generated by the command is sensitive and not read-only, which are indicated by the Sensitive and ReadOnly properties of the EDBCommand class, respectively.

In addition, you can set an expression filter on the cursor using the Filter and Filtered properties, and search for a particular row in the cursor using the Locate method. Filter expressions can be any valid Boolean SQL expression that does not contain any sub-queries.

> **Note**
> In addition to the normal Set* methods that use column indexes to set column data, there are also matching Set* methods that accept a column name instead of a column index. The only difference in the method parameters for these methods is the first parameter, which is a String instead of an Int32.

Namespace: Elevate.ElevateDB.Data

Inherits From Elevate.ElevateDB.Data.EDBDataReader

Constructors

All constructors are the same as the EDBDataReader class.

Properties

Property	Description
State: EDBDataCursorState Enumeration	Indicates the state of the cursor.
BOF: Boolean	Indicates whether the cursor is at the beginning of the set of rows that comprise the result set.
EOF: Boolean	Indicates whether the cursor is at the end of the set of rows that comprise the result set.
RowLocked: Boolean	Use this property to determine if the current row has been locked by the LockCurrentRow method. This method only includes manually-locked rows and will not indicate if a row is locked via the Update method when the current session's row lock protocol is set to lpPessimistic. Such row locks are considered implicit.
Filter: String	Use this property to specify a Boolean filter expression for limiting the view of the result set to only those rows that match the filter expression. Setting this property by itself does nothing. You must set the Filtered property in order to actually turn on or off the filtering.
Filtered: Boolean	Use this property turn on or off the filter specified by the Filter property.

RowCount: Integer	Indicates the number of rows present in the current view of the result set. If the result set is filtered via the Filter and Filtered properties, then this property will reflect only the count of the rows that satisfy the filter expression.

Methods

Method	Description
get_Filter: String	This is the accessor (getter) method for the Filter property.
set_Filter(Value: String)	This is the mutator (setter) method for the Filter property.
get_Filtered: Boolean	This is the accessor (getter) method for the Filtered property.
set_Filtered(Value: Boolean)	This is the mutator (setter) method for the Filtered property.
get_RowCount: Integer	This is the accessor (getter) method for the RowCount property.
ReadFirst: Boolean	Use this method to position the cursor on the first row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the first row that satisfies the filter expression. This method always sets the BOF property to True.
ReadLast: Boolean	Use this method to position the cursor on the last row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the last row that satisfies the filter expression. This method always sets the EOF property to True.
Read: Boolean	Use this method to position the cursor on the next row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the next row that satisfies the filter expression. If there are no more rows in the result set, then this method will set the EOF property to True and the existing row buffer will stay the same.
ReadPrevious: Boolean	Use this method to position the cursor on the previous row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the previous row that satisfies the filter expression. If there are no prior rows in the result set, then this method will set the BOF property to True and the existing row buffer will stay the same.
ReadCurrent: Boolean	Use this method to read the current row in the result set into the row buffer. If there are no rows in the result set, then this method will set the BOF and EOF properties to True and the row buffer will contain NULL values for all columns.
ReadRelative(Position: Integer): Boolean	Use this method to position the cursor on a specific row in relation to the current row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the specific row that satisfies the filter expression.

	The Position parameter can be either positive or negative. If it is positive, then the cursor will skip forward Position number of rows from the current row. If there are no more rows in the result set, then this method will set the EOF property to True and the existing row buffer will be populated with the last accessible row that was visited. If it is negative, then the cursor will skip backward Position number of rows from the current row. If there are no prior rows in the result set, then this method will set the BOF property to True and the existing row buffer will be populated with the last accessible row that was visited.
ReadAbsolute(Position: Integer): Boolean	Use this method to position the cursor on a specific row in the result set and read it into the row buffer. If a filter is in effect on the result set, then this method will position the cursor on the specific row that satisfies the filter expression. The Position parameter must be positive. If the Position is greater than the number of rows in the result set, then this method will set the EOF property to True and the existing row buffer will be populated with the last accessible row.
LockCurrentRow	Use this method to manually lock the current row. Row locks established via this method are persistent and are maintained across any Update or Delete method calls. You must manually unlock any rows locked using this method via the UnlockCurrentRow or UnlockAllRows methods. **Note** Any row locks established using this method are automatically unlocked when the cursor is closed.
UnlockCurrentRow	Use this method to unlock the current row. The current row must have been previously manually locked using the LockCurrentRow method or else an exception will be raised.
UnlockAllRows	Use this method to unlock all rows that have been manually locked using the LockCurrentRow method.
Insert	Use this method to put the cursor in an Inserting state. All column values in the row buffer will be set to their default value as defined for the source table used to generate the result set. To complete the insertion of a new row, use the Set* methods to assign new values to the columns in the row buffer, and then call the Post method. If there are any errors in the insert, they will occur during the Post method call, and one can use a try..catch/except block to handle any exceptions generated due to an error such as a constraint violation.
Update	Use this method to put the cursor in an Updating state. If the row locking protocol in use is the pessimistic protocol, then calling this method will cause a row lock to be placed on the current row in the result set. If the row locking protocol in use is the optimistic protocol, then calling this method will refresh

	the row buffer from the current row in the result set. Please see the EDBConnectionStringBuilder Class for information on setting the row lock protocol for the connection. To complete the update of the current row, use the Set* methods to assign new values to the columns in the row buffer, and then call the Post method. If there are any errors in the update, they will occur during the Post method call, and one can use a try..catch/except block to handle any exceptions generated due to an error such as a constraint violation. If the Post method succeeds, then any pessimistic row lock obtained during the Update method call will be released. If optimistic row locking is being used, then a row lock will be obtained and released all within the Post method call.
Post	Use this method to complete an insert or update by posting any changes to the row buffer into the result set. If the State of the cursor is Inserting, then calling this method will insert a new row into the result set. If the State of the cursor is Updating, then calling this method will update the current row in the result set.
Cancel	Use this method to cancel an insert or update, discarding any changes to the row buffer that have been made. If the State of the cursor is Updating and pessimistic locking is being used, then this method will release any row lock that was obtained during the prior Update call.
Delete	Use this method to delete the current row from the result set.
Locate(const Columns: array of String; const Values: ObjectArray; PartialLength: Integer; CaseInsensitive: Boolean): Boolean Locate(const Column: String; const Value: Object; PartialLength: Integer; CaseInsensitive: Boolean): Boolean	Use this method to locate a specific row in the result set. Specify the columns to search using the Columns array parameter and the Values to search for in the Values array parameter. You can specify a partial length for the last string value in the Values parameter via the PartialLength parameter. If you don't want a partial-length match on the last value in the Values parameter, then simply specify 0 for th PartialLength parameter. If you want the search to be case-insensitive, then specify true/True for the CaseInsensitive parameter. The Locate method will respect any filter that may be present on the result set.
SetValue(Index: Int32; Value: System.Object)	Use this method to set a column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetValues(Values: ObjectArray): Int32	Use this method to set multiple column values in the row buffer in one call. The values are assigned starting from the first column to the last column that corresponds to the length of the Values parameter.

SetBoolean(Index: Int32; Value: Boolean)	Use this method to set a Boolean column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetByte(Index: Int32; Value: System.Byte)	This method is not supported by ElevateDB, but is included for compatibility with other cursor classes for .Net.
SetBytes(Index: Int32; Offset: Int64; Buffer: array of System.Byte; BufferOffset: Int32; Length: Int32) SetBytes(Index: Int32; SourceStream: Stream): Int64 SetBytes(Index: Int32; const SourceFileName: String): Int64	Use these methods to set a CHAR/VARCHAR, BYTES/VARBYTES, CLOB, or BLOB column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetImage(Index: Int32; SourceImage: Bitmap; SourceFormat: ImageFormat)	Use this method to save a bitmap to a BLOB column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetChar(Index: Int32; Value: System.Char)	This method is not supported by ElevateDB, but is included for compatibility with other cursor classes for .Net.
SetChars(Index: Int32; Offset: Int64; Buffer: array of System.Char; BufferOffset: Int32; Length: Int32)	Use this method to set a specific range of characters for a CLOB or character (CHAR/VARCHAR) column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetGuid(Index: Int32; Value: System.Guid)	Use this method to set a GUID column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetInt16(Index: Int32; Value: Int16)	Use this method to set a signed 16-bit integer column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetInt32(Index: Int32; Value: Int32)	Use this method to set a signed 32-bit integer column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetInt64(Index: Int32; Value: Int64)	Use this method to set a signed 64-bit integer column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetFloat(Index: Int32; Value: Single)	This method is not supported by ElevateDB, but is included for compatibility with other cursor classes for .Net.
SetDouble(Index: Int32; Value: System.Double)	Use this method to set a double-precision floating-point column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.

SetString(Index: Int32; Value: String)	Use this method to set a string column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetDecimal(Index: Int32; Value: System.Decimal)	Use this method to set a decimal column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.
SetDateTime(Index: Int32; Value: System.DateTime)	Use this method to set a date/time column value in the row buffer. The column is specified via the Index parameter (0-based), and the cursor must be in the Inserting or Updating state.

Events

Event	Description
OnStateChange: EDBDataCursorStateChangeEvent	Setting this event adds an event handler to the list of event handlers listening for the OnStateChange event for the cursor. The arguments for the event handler are defined in the EDBCursorStateChangeEvent class. The OnStateChange event is used to respond to any change in the cursor's State property.
OnMove: EDBDataCursorMoveEvent	Setting this event adds an event handler to the list of event handlers listening for the OnMove event for the cursor. The arguments for the event handler are defined in the EDBCursorMoveEvent class. The OnMove event is used to respond to any movement in the cursor due to calls to the ReadFirst, ReadLast, Read, ReadPrevious, ReadRelative, ReadAbsolute, or Locate methods.

3.33 EDBDataAdapter Class

The EDBDataAdapter class implements the DbDataAdapter abstract class, providing an encapsulation of a data adapter for executing queries using SELECT statements and updating any result sets using INSERT, UPDATE, and/or DELETE statements.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbDataAdapter

Constructors

(<No Parameters>)
(SelectCommand: EDBCommand)
(SelectCommandText: String; SelectConnectionString: String)
(SelectCommandText: String; SelectConnection: EDBConnection)

Properties

Property	Description
EngineVersion: String	Indicates the current version of ElevateDB being used. This property is read-only.

3.34 EDBCommandBuilder Class

The EDBCommandBuilder class extends the DbCommandBuilder abstract class, used for generating the appropriate commands for reconciling changes back to a dataset.

Namespace: Elevate.ElevateDB.Data

Inherits From System.Data.Common.DbCommandBuilder

Constructors

(<No Parameters>)
(Adapter: EDBDataAdapter)

Methods

Method	Description
DeriveParameters(Command: EDBCommand)	Use this method to populate the parameters for an EDBCommand instance. **Note** This method simply calls the Prepare method of the EDBCommand instance to populate the parameters, so calling this method will cause the EDBCommand instance to be closed (if it is open) and then prepared.

This page intentionally left blank

Appendix A - Error Codes and Messages

The following is a table of the error codes and messages for ElevateDB.

Error Code	Message and Further Details
EDB_ERROR_VALIDATE (100)	There is an error in the metadata for the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever an attempt is made to create a new catalog or configuration object, and there is an error in the specification of the object. The specific error message is indicated within the parentheses.
EDB_ERROR_UPDATE (101)	There was an error updating the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to update the disk file used to store a catalog or configuration. The specific error message is indicated within the parentheses.
EDB_ERROR_SYSTEM (200)	This operation cannot be performed on the system <ObjectType> <ObjectName> or any privileges granted to itThis error is raised whenever an attempt is made to alter or drop any system-defined catalog or configuration objects. Please see the System Information topic for more information on the system-defined objects in ElevateDB.
EDB_ERROR_DEPENDENCY (201)	The <ObjectType> <ObjectName> cannot be dropped or moved because it is still referenced by the <ObjectType> <ObjectName>This error is raised whenever an attempt is made to drop any catalog or configuration object, and that catalog or configuration object is still being referenced by another catalog or configuration object. You must first remove the reference to the object that you wish to drop before you can drop the referenced object.
EDB_ERROR_MODULE (202)	An error occurred with the module <ModuleName> (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue with loading an external module. Please see the External Modules topic for more information.
EDB_ERROR_LOCK (300)	Cannot lock <ObjectType> <ObjectName> for <AccessType> accessThis error is raised whenever ElevateDB cannot obtain the desired lock access to a given catalog or configuration object. This is usually due to another session already having an incompatible lock on the object already. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_UNLOCK (301)	Cannot unlock <ObjectType> <ObjectName> for <AccessType> accessThis error is raised whenever ElevateDB cannot unlock a given catalog or configuration object. If this error occurs during normal operation of ElevateDB, please contact Elevate Software for further

	instructions on how to correct the issue
EDB_ERROR_EXISTS (400)	The <ObjectType> <ObjectName> already existsThis error is raised whenever an attempt is made to create a new catalog or configuration object, and a catalog or configuration object already exists with that name.
EDB_ERROR_NOTFOUND (401)	The <ObjectType> <ObjectName> does not existThis error is raised when an attempt is made to open/execute, alter, or drop a catalog or configuration object that does not exist.
EDB_ERROR_NOTOPEN (402)	The database <DatabaseName> must be open in order to perform this operation (<OperationName>)This error is raised when an attempt is made to perform an operation on a given database before it has been opened.
EDB_ERROR_READONLY (403)	The <ObjectType> <ObjectName> is read-only and this operation cannot be performed (<OperationName>)This error is raised whenever a create, alter, or drop operation is attempted on an object that is read-only.
EDB_ERROR_TRANS (404)	This operation cannot be performed while the database <DatabaseName> has an active transaction (<OperationName>)This error is raised whenever ElevateDB encounters an invalid transaction operation. Some SQL statements cannot be executed within a transaction. For a list of transaction-compatible statements, please see the Transactions topic.
EDB_ERROR_MAXIMUM (405)	The maximum number of <ObjectType>s has been reached (<MaximumObjectsAllowed>)This error is raised when an attempt is made to create a new catalog or configuration object and doing so would exceed the maximum allowable number of objects. Please see the Appendix B - System Capacities topic for more information.
EDB_ERROR_IDENTIFIER (406)	Invalid <ObjectType> identifier '<ObjectName>'This error is raised when an attempt is made to create a new catalog or configuration object with an invalid name. Please see the Identifiers topic for more information on what constitutes a valid identifier.
EDB_ERROR_FULL (407)	The table <TableName> is full (<FileName>)This error occurs when a given table contains the maximum number of rows or the maximum file size is reached for one of the files that make up the table. The file name is indicated within the parentheses.
EDB_ERROR_CONFIG (409)	There is an error in the configuration (<ErrorMessage>)This error is raised whenever there is an error in the configuration. The specific error message is indicated within the parentheses.

EDB_ERROR_NOLOGIN (500)	A user must be logged in in order to perform this operation (<OperationName>)This error is raised whenever an attempt is made to perform an operation for a session that has not been logged in yet with a valid user name and password.
EDB_ERROR_LOGIN (501)	Login failed (<ErrorMessage>)This error is raised whenever a user login fails. ElevateDB allows for a maximum of 3 login attempts before raising a login exception.
EDB_ERROR_ADMIN (502)	Administrator privileges are required to perform this operation (<Operation>)This error is raised when an attempt is made to perform an operation that requires administrator privileges. Administrator privileges are granted to a given user by granting the system-defined "Administrators" role to that user. Please see the User Security topic for more information.
EDB_ERROR_PRIVILEGE (503)	The current user does not have the proper privileges to perform this operation (<OperationName>)This error is raised when a user attempts an operation when he/she does not have the proper privileges required to execute the operation. Please see the User Security topic for more information.
EDB_ERROR_MAXSESSIONS (504)	Maximum number of concurrent sessions reached for the configuration <ConfigurationName>This error is raised when the maximum number of licensed sessions for a given configuration is exceeded. The number of licensed sessions for a given configuration depends upon the ElevateDB product purchased along with the particular compilation of the application made by the developer using the ElevateDB product.
EDB_ERROR_SERVER (505)	The ElevateDB Server cannot be started (<ErrorMessage>) The ElevateDB Server cannot be stopped (<ErrorMessage>)This error is raised when the ElevateDB Server cannot be started or stopped for any reason. Normally, the error message will contain a native operating system error message that will reveal the reason for the issue.
EDB_ERROR_FILEMANAGER (600)	File manager error (<ErrorMessage>)This error is raised whenever ElevateDB encounters a file manager error while trying to create, open, close, delete, or rename a file. The specific error message, including operating system error code (if available), is indicated within the parentheses.
EDB_ERROR_CORRUPT (601)	The table <TableName> is corrupt (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while reading, writing, or validating a table. If this error occurs during normal operation of ElevateDB, please contact Elevate Software for further instructions on how to correct the issue. The specific error message is indicated within the parentheses.

EDB_ERROR_COMPILE (700)	An error was found in the <ObjectType> at line <Line> and column <Column> (<ErrorMessage>)This error is raised whenever an error is encountered while compiling an SQL expression, statement, or routine. The specific error message is indicated within the parentheses.
EDB_ERROR_BINDING (800)	A row binding error occurredThis error is raised when ElevateDB encounters an issue while trying to bind the cursor row values in a cursor row. It is an internal error and will not occur unless there is a bug in ElevateDB.
EDB_ERROR_STATEMENT (900)	An error occurred with the statement <StatementName> (<ErrorMessage>)This error is raised whenever an issue is encountered while executing a statement. The specific error message is indicated within the parentheses.
EDB_ERROR_PROCEDURE (901)	An error occurred with the procedure <ProcedureName> (<ErrorMessage>)This error is raised whenever an issue is encountered while executing a procedure. The specific error message is indicated within the parentheses.
EDB_ERROR_VIEW (902)	An error occurred with the view <ViewName> (<ErrorMessage>)This error is raised whenever an issue is encountered while opening a view. The specific error message is indicated within the parentheses.
EDB_ERROR_JOB (903)	An error occurred with the job <JobName> (<ErrorMessage>)This error is raised whenever an issue is encountered while running a job. The specific error message is indicated within the parentheses.
EDB_ERROR_IMPORT (904)	Error importing the file <FileName> into the table <TableName> (<ErrorMessage>)This error is raised when an error occurs during the import process for a given table. The specific error message is indicated within the parentheses.
EDB_ERROR_EXPORT (905)	Error exporting the table <TableName> to the file <FileName> (<ErrorMessage>)This error is raised when an error occurs during the export process for a given table. The specific error message is indicated within the parentheses.
EDB_ERROR_CURSOR (1000)	An error occurred with the cursor <CursorName> (<ErrorMessage>)This error is raised whenever an issue is encountered while operating on a cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_FILTER (1001)	A filter error occurred (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to set or clear a filter on a given cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_LOCATE (1002)	A locate error occurred (<ErrorMessage>)This error is raised whenever ElevateDB encounters an issue while trying to locate a row in a given cursor. The specific error message is indicated within the parentheses.

EDB_ERROR_STREAM (1003)	An error occurred in the cursor stream (<ErrorMessage>)This error is raised whenever an issue is encountered while loading or saving a cursor to or from a stream. The specific error message is indicated within the parentheses.
EDB_ERROR_CONSTRAINT (1004)	The constraint <ConstrainName> has been violated (<ErrorMessage>)This error is raised when a constraint that has been defined for a table is violated. This includes primary key, unique key, foreign key, and check constraints. The specific error message is indicated within the parentheses.
EDB_ERROR_LOCKROW (1005)	Cannot lock the row in the table <TableName>This error is raised when a request is made to lock a given row and the request fails because another session has the row already locked. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_UNLOCKROW (1006)	Cannot unlock the row in the table <TableName>This error is raised whenever ElevateDB cannot unlock a specific row because the row had not been previously locked, or had been locked and the lock has since been cleared. Please see the Locking and Concurrency topic for more information.
EDB_ERROR_ROWDELETED (1007)	The row has been deleted since last cached for the table <TableName>This error is raised whenever an attempt is made to update or delete a row, and the row no longer exists because it has been deleted by another session. Please see the Updating Rows topic for more information.
EDB_ERROR_ROWMODIFIED (1008)	The row has been modified since last cached for the table <TableName>This error is raised whenever an attempt is made to update or delete a row, and the row has been updated by another session since the last time it was cached by the current session. Please see the Updating Rows topic for more information.
EDB_ERROR_CONSTRAINED (1009)	The cursor is constrained and this row violates the current cursor constraint condition(s)This error is raised when an attempt is made to insert a new row into a constrained cursor that violates the filter constraints defined for the cursor. Both views defined in database catalogs and the result sets of dynamic queries can be defined as constrained, and the filter constraints in both cases are the WHERE conditions defined for the underlying SELECT query that the view or dynamic query is based upon.
EDB_ERROR_ROWVISIBILITY (1010)	The row is no longer visible in the table <TableName>This error is raised whenever an attempt is made to update or delete a row within the context of a cursor with an active filter or range condition, and the row has been updated by another session since the last time it was cached by the current session, thus causing it to fall out of the scope of the cursor's active filter or range condition. Please see the Updating Rows topic for

	more information.
EDB_ERROR_VALUE (1011)	An error occurred with the <ObjectType> <ObjectName> (<ErrorMessage>)This error is raised whenever an attempt is made to store a value in a column, parameter, or variable and the value is invalid because it is out of range or would be truncated. The specific error message is indicated within the parentheses.
EDB_ERROR_CLIENTCONN (1100)	A connection to the server at <ServerAddress> cannot be established (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while trying to connect to a remote ElevateDB Server. The error message will indicate the reason why the connection cannot be completed.
EDB_ERROR_CLIENTLOST (1101)	A connection to the server at <ServerAddress> has been lost <ErrorMessage>)This error is raised when ElevateDB encounters an issue while connected to a remote ElevateDB Server. The error message will indicate the reason why the connection was lost.
EDB_ERROR_INVREQUEST (1103)	An invalid or unknown request was sent to the serverThis error is raised when an ElevateDB Server encounters an unknown request from a client session.
EDB_ERROR_ADDRBLOCK (1104)	The IP address <IPAddress> is blockedThis error is raised when a session tries to connect to an ElevateDB Server, and the originating IP address for the session matches one of the configured blocked IP addresses in the ElevateDB Server, or does not match one of the configured authorized IP addresses in the ElevateDB Server.
EDB_ERROR_ENCRYPTREQ (1105)	An encrypted connection is requiredThis error is raised when a non-encrypted session tries to connect to an ElevateDB Server that has been configured to only accept encrypted session connections.
EDB_ERROR_SESSIONNOTFOUND (1107)	The session ID <SessionID> is no longer present on the serverThis error is raised whenever a remote session attempts to reconnect to a session that has already been designated as a dead session and removed by the ElevateDB Server. This can occur when a session is inactive for a long period of time, or when the ElevateDB Server has been stopped and then restarted.
EDB_ERROR_SESSIONCURRENT (1108)	The current session ID <SessionID> cannot be disconnected or removedThis error is raised whenever a remote session attempts to disconnect or remove itself.
EDB_ERROR_COMPRESS (1200)	An error occurred while compressing data (<ErrorMessage>)This error is raised when ElevateDB encounters an issue while attempting to compress data. It is an internal error and will not occur unless there is a bug in ElevateDB. The specific error message is indicated within the parentheses.
EDB_ERROR_DECOMPRESS (1201)	An error occurred while uncompressing data

	(<ErrorMessage>)This error is raised when ElevateDB encounters an issue while attempting to decompress data. It is an internal error and will not occur unless there is a bug in ElevateDB. The specific error message is indicated within the parentheses.
EDB_ERROR_BACKUP (1300)	Error backing up the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the backing up of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_RESTORE (1301)	Error restoring the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the restore of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_PUBLISH (1302)	Error publishing the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the publishing of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_UNPUBLISH (1303)	Error unpublishing the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the unpublishing of a database. The specific error message is indicated within the parentheses.
EDB_ERROR_SAVEUPDATES (1304)	Error saving updates for the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the saving of the updates for a database. The specific error message is indicated within the parentheses.
EDB_ERROR_LOADUPDATES (1305)	Error loading updates for the database <DatabaseName> (<ErrorMessage>)This error is raised when any error occurs during the loading of the updates for a database. The specific error message is indicated within the parentheses.
EDB_ERROR_STORE (1306)	Error with the store <StoreName> (<ErrorMessage>)This error is raised when any error occurs while trying to access a store, such as a read or write error while working with files in the store. The specific error message is indicated within the parentheses.
EDB_ERROR_CACHEUPDATES (1307)	Error caching updates for the cursor <CursorName> (<ErrorMessage>)This error is raised when any error occurs during the caching of updates for a specific table, view, or query cursor. The specific error message is indicated within the parentheses.
EDB_ERROR_FORMAT (1400)	Error in the format string <FormatString> (<ErrorMessage>)This error is raised when ElevateDB encounters an issue with a format string used in a date, time, or timestamp format used in a table import or export. The specific error message is indicated within the parentheses.

This page intentionally left blank

Appendix B - System Capacities

The following is a list of the capacities for the different objects in ElevateDB. Any object that is not specifically mentioned here has an implicit capacity of 2147483647, or High(Integer). For example, there is no stated capacity for the maximum number of roles allowed in a configuration. Therefore, the implicit capacity is 2147483647 roles.

Capacity	Details
Max BLOB Column Size	The maximum size of a BLOB column is 2GB.
Max CHAR/VARCHAR Column Length	The maximum length of a VARCHAR/CHAR columns is 1024 characters.
Max Identifier Length	The maximum length of an identifier is 80 characters.
Max Number of Columns in a Table	The maximum number of columns in a table is 2048.
Max Number of Columns in an Index	The maximum number of columns in an index is limited by the table's defined index page size.
Max Number of Concurrent Sessions	The maximum number of concurrent sessions for an application or ElevateDB server is 4096.
Max Number of Indexes in a Table	The maximum number of indexes in a table is 512.
Max Number of Jobs in a Configuration	The maximum number of jobs in a configuration is 4096.
Max Number of Routines in a Database	The maximum number of routines (procedures and functions combined) in a database is 4096.
Max Number of Rows in a Table	The maximum number of rows in a table is determined by whether global file I/O buffering is enabled in ElevateDB. If global file I/O buffering is enabled, then the maximum number of rows is determined by the maximum file size permitted in the operating system. If global file I/O buffering is not enabled, then the approximate maximum number of rows can be determined by dividing 128GB by the row size.
Max Number of Rows in a Transaction	The maximum number of rows in a single transaction is only limited by the available memory constraints of the operating system and/or hardware.
Max Number of Tables in a Database	The maximum number of tables in a database is 4096.
Max Number of Users in a Configuration	The maximum number of users in a configuration is 4096.
Max Row Size for a Table	The maximum row size for a table is 2GB.
Max Scale for DECIMAL/NUMERIC Columns	The maximum scale for DECIMAL or NUMERIC columns is 4.
Max Size of an In-Memory Table	The maximum size of an in-memory table is only limited by the available memory constraints of the operating system or hardware.
Min/Max BLOB Block Size for a Table	The minimum BLOB block size is 64 bytes for ANSI databases and 128 bytes for Unicode databases. The maximum BLOB block size is 2GB.

Min/Max Index Page Size for a Table	The minimum index page size is 1 kilobyte for ANSI databases and 2 kilobytes for Unicode databases. The maximum index page size is 2GB.