

Contents

Chapter 1

Namespace

FirebirdSql.Data.Firebird

Namespace Contents

Page

Interfaces

Classes

FbCommand??	??
<i>Represents an SQL statement or stored procedure to execute against a data source. This class cannot be inherited.</i>	
FbCommandBuilder??	??
<i>Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated data source. This class cannot be inherited.</i>	
FbConnection??	??
<i>Represents an open connection to a Firebird database. This class cannot be inherited.</i>	
FbDataAdapter??	??
<i>Represents a set of data commands and a connection to a data source that are used to fill the DataSet and update the data source. This class cannot be inherited.</i>	
FbDatabaseInfo??	??
<i>Reports requested information about a attached database. This class cannot be inherited.</i>	
FbDataReader??	??
<i>Provides a Way for reading a forward-only stream of rows. This class cannot be inherited.</i>	
FbError??	??
<i>Collects information relevant to a error returned by Firebird. This class cannot be inherited.</i>	
FbErrorCollection??	??
<i>Collects all errors generated by the Firebird .NET Data Provider. This class cannot be inherited.</i>	

FbException	??
<i>The exception that is thrown when Firebird Server returns a warning or error. This class cannot be inherited.</i>	
FbInfoMessageEventArgs	??
<i>Provides data for the InfoMessage event. This class cannot be inherited.</i>	
FbParameter	??
<i>Represents a parameter of a , and optionally, its mapping to DataSet columns. This class cannot be inherited.</i>	
FbParameterCollection	??
<i>A collection of parameters associated to a . This class cannot be inherited.</i>	
FbRowUpdatedEventArgs	??
<i>Provides data for the RowUpdated event. This class cannot be inherited.</i>	
FbRowUpdatingEventArgs	??
<i>Provides data for the RowUpdating event. This class cannot be inherited.</i>	
FbTransaction	??
<i>Represents a Firebird transaction. This class cannot be inherited.</i>	

1.1 Interfaces

1.2 Classes

1.2.1 CLASS FbCommand

Represents an SQL statement or stored procedure to execute against a data source. This class cannot be inherited. The FbCommand class provides the following methods for executing commands against a Firebird database:

MethodDescriptionExecuteReaderExecutes commands that return rows.
ExecuteNonQueryExecutes commands such as SQL INSERT, DELETE, UPDATE, and SET statements.
ExecuteScalarRetrieves a single value (for example, an aggregate value) from a database.

DECLARATION

```
public class FbCommand
    : Component
```

PROPERTIES

- *CommandPlan*

```
public string CommandPlan { get; }
```

Gets the plan used by the server for the command.

– **Usage**

- * It's needed that the command was prepared to retrieve the command plan.

- *CommandText*

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

Gets or sets the SQL statement or stored procedure to execute.

– **Usage**

- * When the property is set to StoredProcedure, the CommandText property should be set to the name of the stored procedure. The user may be required to use escape character syntax if the stored procedure name contains any special characters. The command

executes this stored procedure when you call one of the Execute methods.

The Firebird .NET Data Provider support the question mark (?) placeholder and named parameters for passing parameters to a SQL Statement or a stored procedure.

For example you can do:

```
SELECT * FROM Customers WHERE CustomerID =
@CustomerID
```

or

```
SELECT * FROM Customers WHERE CustomerID = ?
```

- *CommandTimeout*

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

Gets or sets the wait time before cancel command execution and generating an error.

- **Usage**

- * A value of 0 indicates no limit, and should be avoided in a CommandTimeout because an attempt to execute a command will wait indefinitely. Not currently supported.

- *CommandType*

```
public System.Data.CommandType CommandType { get; set; }
}
```

Gets or sets how the property is interpreted.

- *Connection*

```
public FirebirdSql.Data.Firebird.FbConnection Connection {
get; set; }
```

Gets or sets the associated with the current command.

- **Usage**

- * You cannot set the Connection, , and properties if the current connection is performing an execute or fetch operation.

- *Container*

```
public System.ComponentModel.IContainer Container { get; }
```

- *CursorName*

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

Gets or sets the cursor name for the object.

- *DesignMode*

```
protected bool DesignMode { get; }
```

- *DesignTimeVisible*

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

Gets or sets whether the command object should be visible in a Windows Forms Designer control.

- *Events*

```
protected System.ComponentModel.EventHandlerList Events {  
get; }
```

- *Parameters*

```
public FirebirdSql.Data.Firebird.FbParameterCollection  
Parameters { get; }
```

Gets the .

- *Site*

```
public System.ComponentModel.ISite Site { get; set; }
```

- *Transaction*

```
public FirebirdSql.Data.Firebird.FbTransaction Transaction {  
get; set; }
```

Gets or sets the within which the will be used.

- **Usage**

- * You cannot set the Transaction property if it is already set to a specific value, and the command is in the process of executing. If you set the transaction property to an object that is different of the of the object, an exception will be thrown the next time you attempt to execute or prepare the command.

- *UpdatedRowSource*

```
public System.Data.UpdateRowSource UpdatedRowSource {
    get; set; }
```

Gets or sets how the Update method should apply command results to the DataRow.

– **Usage**

- * The default UpdateRowSource value is Both unless the command is automatically generated, for example using the `ExecuteNonQuery()`, in which case the default will be None.

CONSTRUCTORS

- *.ctor*

```
public FbCommand( )
```

Creates a new instance of the class.

- *.ctor*

```
public FbCommand( )
```

Creates a new instance of the class with the text of the query.

– **Parameters**

- * `cmdText` -

- *.ctor*

```
public FbCommand( )
```

– **Parameters**

- * `cmdText` -
- * `connection` -

- *.ctor*

```
public FbCommand( )
```

– **Parameters**

- * cmdText -
- * connection -
- * transaction -

METHODS

- *Cancel*

public void Cancel()

Try to cancel command execution.

- **Usage**

- * Not currently supported.

- *CreateObjRef*

public System.Runtime.Remoting.ObjRef CreateObjRef()

- **Parameters**

- * requestedType -

- *CreateParameter*

public FirebirdSql.Data.Firebird.FbParameter CreateParameter()

Creates a new instance of an class.

- *Dispose*

public void Dispose()

- *Dispose*

protected void Dispose()

Releases the unmanaged and, optionally, the managed resources used by the object.

- **Parameters**

- * disposing -

- *Equals*

```
public bool Equals( )
```

- **Parameters**

- * obj -

- *ExecuteNonQuery*

```
public int ExecuteNonQuery( )
```

Executes an the query and returns the number of rows affected.

- **Usage**

- * Although ExecuteNonQuery does not return any rows, any output parameters or return values mapped to parameters are populated with data. **IMPORTANT:** To execute this method you need to have a associated to the command.

- *ExecuteReader*

```
public FirebirdSql.Data.Firebird.FbDataReader ExecuteReader( )
```

Executes the query and builds an object.

- *ExecuteReader*

```
public FirebirdSql.Data.Firebird.FbDataReader ExecuteReader( )
```

Executes the query and builds an FbDataReader using one of the CommandBehavior values.

- **Parameters**

- * behavior -

- *ExecuteScalar*

```
public object ExecuteScalar( )
```

Executes the query, and returns the first column of the first row in the resultset returned by the query. Extra columns or rows are ignored.

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetService*

```
protected object GetService( )
```

– Parameters

* service -

- *GetType*

```
public System.Type GetType( )
```

- *ICloneable.Clone*

```
private object ICloneable.Clone( )
```

- *IDbCommand.CreateParameter*

```
private System.Data.IDbDataParameter  
IDbCommand.CreateParameter( )
```

- *IDbCommand.ExecuteReader*

```
private System.Data.IDataReader IDbCommand.ExecuteReader(  
)
```

– Parameters

* behavior -

- *IDbCommand.ExecuteReader*

```
private System.Data.IDataReader IDbCommand.ExecuteReader(  
    )
```

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Prepare*

```
public void Prepare( )
```

Creates a prepared (or compiled) version of the command.

– **Usage**

- * Prepare is automatically called when the command is executed using `Execute`, `ExecuteReader`, or `ExecuteNonQuery` methods.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.2 CLASS FbCommandBuilder

Automatically generates single-table commands used to reconcile changes made to a DataSet with the associated data source. This class cannot be inherited. The does not automatically generate the SQL statements required to reconcile changes made to a DataSet associated with the data source. However, you can create an FbCommandBuilder object that generates SQL statements for single-table updates by setting the SelectCommand property of the . Then, the FbCommandBuilder generates any additional SQL statements that you do not set.

To generate INSERT, UPDATE, or DELETE statements, the FbCommandBuilder uses the property to retrieve a required set of metadata. If you change the value of after the metadata has been retrieved (for example, after the first update), you then should call the method to update the metadata.

DECLARATION

```
public class FbCommandBuilder
    : Component
```

PROPERTIES

- *Container*

```
public System.ComponentModel.IContainer Container { get; }
```

- *DataAdapter*

```
public FirebirdSql.Data.Firebird.FbDataAdapter DataAdapter {
    get; set; }
```

Gets or sets an object for which this object will generate SQL statements.

– **Usage**

- * The registers itself as a listener for events that are generated by the specified in this property.

- *DesignMode*

```
protected bool DesignMode { get; }
```

- *Events*

```
protected System.ComponentModel.EventHandlerList Events {
    get; }
```

- *QuotePrefix*

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

Gets or sets the beginning character or characters to use when working with database objects (for example, tables or columns) whose names contain characters such as spaces or reserved tokens.

- **Usage**

- * Some data sources may contain objects whose names include characters such as spaces, commas, and semicolons. To accommodate this, use the `QuotePrefix` and `QuoteSuffix` properties to specify delimiters, such as a left and right bracket, that will encapsulate the object name. Note: Although you cannot change the `QuotePrefix` or `QuoteSuffix` properties after an insert, update, or delete command has been generated, you can change their settings after calling the `Update` method of an `SqlCommand`.

- *QuoteSuffix*

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

Gets or sets the ending character or characters to use when working with database objects, (for example, tables or columns), whose names contain characters such as spaces or reserved tokens.

- **Usage**

- * Some data sources may contain objects whose names include characters such as spaces, commas, and semicolons. To accommodate this, use the `QuotePrefix` and `QuoteSuffix` properties to specify delimiters, such as a left and right bracket, that will encapsulate the object name. Note: Although you cannot change the `QuotePrefix` or `QuoteSuffix` properties after an insert, update, or delete command has been generated, you can change their settings after calling the `Update` method of an `SqlCommand`.

- *Site*

```
public System.ComponentModel.ISite Site { get; set; }
```

CONSTRUCTORS

- *.ctor*

```
public FbCommandBuilder( )
```

Creates a new instance of the class.

- *.ctor*

```
public FbCommandBuilder( )
```

– **Parameters**

* adapter -

METHODS

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

– **Parameters**

* requestedType -

- *DeriveParameters*

```
public void DeriveParameters( )
```

– **Parameters**

* command -

- *Dispose*

```
public void Dispose( )
```

- *Dispose*

```
protected void Dispose( )
```

Releases the unmanaged and, optionally, the managed resources used by the object.

– **Parameters**

* disposing -

- *Equals*

```
public bool Equals( )
```

- **Parameters**

- * obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetDeleteCommand*

```
public FirebirdSql.Data.Firebird.FbCommand  
GetDeleteCommand( )
```

Gets the automatically generated object required to perform deletions at the data source.

- **Usage**

- * You can use the GetDeleteCommand method for informational or troubleshooting purposes because it returns the object to be executed.

You can also use GetDeleteCommand as the basis of a modified command. For example, you might call GetDeleteCommand and modify the value, and then explicitly set that on the .

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

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetInsertCommand*

```
public FirebirdSql.Data.Firebird.FbCommand  
GetInsertCommand( )
```

Gets the automatically generated object required to perform insertions at the data source.

– Usage

- * You can use the `GetInsertCommand` method for informational or troubleshooting purposes because it returns the object to be executed. You can also use `GetInsertCommand` as the basis of a modified command.

For example, you might call `GetInsertCommand` and modify the value, and then explicitly set that on the .

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

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetService*

```
protected object GetService( )
```

– Parameters

- * `service` -

- *GetType*

```
public System.Type GetType( )
```

- *GetUpdateCommand*

```
public FirebirdSql.Data.Firebird.FbCommand
GetUpdateCommand( )
```

Gets the automatically generated object required to perform updates at the data source.

– Usage

- * You can use the `GetUpdateCommand` method for informational or troubleshooting purposes because it returns the object to be executed. You can also use `GetUpdateCommand` as the basis of a modified command.

For example, you might call `GetUpdateCommand` and modify the value, and then explicitly set that on the .

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

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *RefreshSchema*

```
public void RefreshSchema( )
```

Refreshes the database schema information used to generate INSERT, UPDATE, or DELETE statements.

– **Usage**

* You should call `RefreshSchema` whenever the value of the changes.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

1.2.3 CLASS FbConnection

Represents an open connection to a Firebird database. This class cannot be inherited. A FbConnection object represents a individual connection to Firebird server.If a FbConnection object goes out of scope, it is not closed, you must explicitly close the the connection by calling or .

DECLARATION

```
public class FbConnection
    : Component
```

PROPERTIES

- *ConnectionString*

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

Gets or sets the string used to open a connection to a Firebird database.

– Usage

- * The following table lists the valid names for keyword values for ConnectionString property.

Name	Description	Default	Database	Database path to establish the connection.
User	Firebird User account for login.	Password	Password	for the Firebird user account.
Dialect	Database dialect.	3	Server or DataSource.	Server name for establish the connection.
localhostPort	Port number in the server for establish the connection.	3050	Charset	Database Character Set.
NONERole	User Role.	Packet Size	Size (in bytes) of network packets used to communicate with an instance of Firebird Server.	8192
Connection Lifetime	When a connection is returned to the pool, its creation time is compared with the current time, and the connection is destroyed if that time span (in seconds) exceeds the value specified by connection lifetime.	0	Pooling	When true, the FbConnection object is drawn from the appropriate pool, or if necessary, is created and added to the appropriate pool. Recognized values are true, false, yes, and no.
true	The following table lists the valid names for the Charset keyword of the ConnectionString:	Firebird	Charset	Description
ASCII	American Standard Code for Information Interchange.	BIG_5	Big5	Traditional Chinese.
DOS437	MS-DOS United States, Australia, New Zealand, South Africa.	DOS850	MS-DOS	Latin-1.
DOS860	MS-DOS			

Portugues.DOS861MS-DOS Icelandic.DOS863MS-DOS Canadian
 French.DOS865MS-DOS Nordic.EUCJ.0208JIS X 0201, 0208, 0212,
 EUC encoding, Japanese.GB.2312GB2312, EUC encoding,
 Simplified Chinese.ISO8859_1ISO 8859-1, Latin alphabet No.
 1.ISO8859_2ISO 8859-2, Latin alphabet No. 2.KSC.5601Windows
 Korean.ISO2022-JPWindows Japanese.SJIS.0208Japanese
 (Shift-JIS)UNICODE_FSSEight-bit Unicode Transformation
 Format.WIN1250Windows Eastern European.WIN1251Windows
 Cyrillic.WIN1252Windows Latin-1.WIN1253Windows
 Greek.WIN1254Windows Turkish.WIN1254Windows
 Hebrew.ArabicWindows Turkish.WIN1257Windows Baltic.

- *ConnectionTimeout*

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

Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

- **Usage**

- * A value of 0 indicates no limit. Not currently supported.

- *Container*

```
public System.ComponentModel.IContainer Container { get; }
```

- *Database*

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

Gets the name of the actual database or the database to be used when a connection is open.

- *DataSource*

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

Gets the name of the Firebird Server to which to connect.

- *DesignMode*

```
protected bool DesignMode { get; }
```

- *Events*

```
protected System.ComponentModel.EventHandlerList Events {
    get; }
```

- *PacketSize*

```
public int PacketSize { get; }
```

Gets the size (in bytes) of network packets used to communicate with an instance of Firebird Server.

- **Usage**

- * PacketSize may be in the range 512-32767 bytes. An exception is generated if the value is outside of this range.

- *ServerVersion*

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

Gets a string containing the version of the Firebird Server to which the client is connected.

- *Site*

```
public System.ComponentModel.ISite Site { get; set; }
```

- *State*

```
public System.Data.ConnectionState State { get; }
```

Gets the current state of the connection.

- **Usage**

- * The allowed state changes are: From Closed to Open, using the method.
From Open to Closed, using the method or the method.

CONSTRUCTORS

- *.ctor*

```
public FbConnection( )
```

Creates a new instance of the class.

- *.ctor*

```
public FbConnection( )
```

Creates a new instance of the class with the specified connection string.

– **Parameters**

* *connString* -

METHODS

- *BeginTransaction*

```
public FirebirdSql.Data.Firebird.FbTransaction
BeginTransaction( )
```

Begins a new transaction with the default isolation level
IsolationLevel.ReadCommitted.

– **Usage**

* To commit or rollback the transaction, you must explicitly use the
or methods.

– **Example**

```
public void RunFirebirdTransaction(string connectionString)
{ FbConnection connection = new
FbConnection(connectionString); connection.Open();
// Start a new transaction FbTransaction transaction =
connection.BeginTransaction();
// Creates a new FbCommand object FbCommand command = new
FbCommand(); command.Connection = connection;
command.Transaction = transaction;
try { command.CommandText = "Insert into Region (RegionID,
RegionDescription) VALUES (100, 'Description')";
command.ExecuteNonQuery(); command.CommandText = "Insert
into Region (RegionID, RegionDescription) VALUES (101,
'Description')"; command.ExecuteNonQuery();
transaction.Commit();
Console.WriteLine("Both records are written to database.");
} catch(Exception e) { myTrans.Rollback();
Console.WriteLine(e.ToString()); Console.WriteLine("Neither
record was written to database."); } finally {
myConnection.Close(); } }
```

- *BeginTransaction*

```
public FirebirdSql.Data.Firebird.FbTransaction
BeginTransaction( )
```

Begins a new transaction with the specified name.

- **Usage**

- * To commit or rollback the transaction, you must explicitly use the `Commit()` or `Rollback()` methods.

- **Parameters**

- * `transactionName` -

- **Example**

```
FbConnection connection = new
FbConnection(connectionString); connection.Open();
// Start a new transaction FbTransaction transaction =
myConnection.BeginTransaction();
// Creates a new FbCommand object FbCommand command = new
FbCommand(); command.Connection = connection;
command.Transaction = transaction;
try { command.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBNP', '.Net Provider', 'N/A')";
command.ExecuteNonQuery();
transaction.Save("SampleTransaction");
command.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBN1', '.Net Provider1.',
'N/A')"; command.ExecuteNonQuery();
transaction.Commit();
Console.WriteLine("Both records are written to database.");
} catch(Exception e) { try {
transaction.Rollback("SampleTransaction"); } catch
(FbException ex) { if (transaction.Connection != null) {
Console.WriteLine("An exception of type " + ex.GetType() + "
was encountered while attempting to roll back the
transaction."); } }
Console.WriteLine("An exception of type " + e.GetType() + "
was encountered while inserting the data.");
Console.WriteLine("Neither record was written to
database."); } finally { connection.Close(); }
```

- *BeginTransaction*

```
public FirebirdSql.Data.Firebird.FbTransaction
BeginTransaction( )
```

Begins a new transaction with the specified isolation level.

– **Usage**

* To commit or rollback the transaction, you must explicitly use the `Commit()` or `Rollback()` methods.

– **Parameters**

* `level` -

– **Example**

```
FbConnection connection = new
FbConnection(connectionString); connection.Open();
FbConnection connection = new
FbConnection(connectionString); connection.Open();
// Start a new transaction FbTransaction transaction =
myConnection.BeginTransaction();
// Creates a new FbCommand object FbCommand command = new
FbCommand(); command.Connection = connection;
command.Transaction = transaction;
try { command.CommandText = "Insert into Region (RegionID,
RegionDescription) VALUES (100, 'Description')";
command.ExecuteNonQuery();
command.CommandText = "Insert into Region (RegionID,
RegionDescription) VALUES (101, 'Description')";
command.ExecuteNonQuery();
transaction.Commit();
Console.WriteLine("Both records are written to database.");
} catch (Exception e) { transaction.Rollback();
Console.WriteLine(e.ToString()); Console.WriteLine("Neither
record was written to database."); } finally {
connection.Close(); }
```

• *BeginTransaction*

```
public FirebirdSql.Data.Firebird.FbTransaction
BeginTransaction( )
```

Begins a new transaction with the specified isolation level and transaction name.

– **Usage**

* To commit or rollback the transaction, you must explicitly use the `Commit()` or `Rollback()` methods.

– **Parameters**

* `level` -

* `transactionName` -

– **Example**

```

FbConnection connection = new
FbConnection(connectionString); connection.Open();
// Start a new transaction FbTransaction transaction =
myConnection.BeginTransaction();
// Creates a new FbCommand object FbCommand command = new
FbCommand(); command.Connection = connection;
command.Transaction = transaction;
try { command.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBNP', '.Net Provider', 'N/A')";
command.ExecuteNonQuery();
transaction.Save("SampleTransaction");
command.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBN1', '.Net Provider1.',
'N/A')"; command.ExecuteNonQuery();
transaction.Commit(); Console.WriteLine("Both records are
written to database."); } catch(Exception e) { try {
transaction.Rollback("SampleTransaction"); } catch
(FbException ex) { if (transaction.Connection != null) {
Console.WriteLine("An exception of type " + ex.GetType() + "
was encountered while attempting to roll back the
transaction."); } }
Console.WriteLine("An exception of type " + e.GetType() + "
was encountered while inserting the data.");
Console.WriteLine("Neither record was written to
database."); } finally { connection.Close(); }

```

• *ChangeDatabase*

```
public void ChangeDatabase( )
```

– **Parameters**

* db -

• *Close*

```
public void Close( )
```

Closes the connection to the database. This is the preferred method of closing any open connection.

– **Usage**

* The method rolls back any pending transactions and drops active statements. It then releases the connection to the connection pool,

or closes the connection if connection pooling is disabled. If Close is called while handling a event, no additional events are fired. can be called more than one time without generating an exception.

- *CreateCommand*

```
public FirebirdSql.Data.Firebird.FbCommand CreateCommand(
)
```

Creates and returns a new object associated with the current object.

- *CreateDatabase*

```
public void CreateDatabase( )
```

- Parameters

- * dataSource -
- * port -
- * database -
- * user -
- * password -
- * dialect -
- * forceWrite -
- * pageSize -
- * charset -

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

- Parameters

- * requestedType -

- *Dispose*

```
public void Dispose( )
```

- *Dispose*

```
protected void Dispose( )
```

Releases the unmanaged and, optionally, the managed resources used by the object.

- Usage

* This method is called by the public Dispose() method and the Finalize method. Dispose() invokes the protected Dispose(Boolean) method with the disposing parameter set to true. Finalize invokes Dispose with disposing set to false.

When the disposing parameter is true, the method releases all resources held by any managed objects that this FbConnection references. It does this by invoking the Dispose() method of each referenced object.

– **Parameters**

* disposing -

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetDbSchemaTable*

```
public System.Data.DataTable GetDbSchemaTable( )
```

Returns information about a specific database schema type using one of the values and the specified restriction values.

– **Parameters**

* schema -

* restrictions -

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetService*

```
protected object GetService( )
```

– Parameters

* service -

- *GetType*

```
public System.Type GetType( )
```

- *ICloneable.Clone*

```
private object ICloneable.Clone( )
```

- *IDbConnection.BeginTransaction*

```
private System.Data.IDbTransaction
IDbConnection.BeginTransaction( )
```

– Parameters

* level -

- *IDbConnection.BeginTransaction*

```
private System.Data.IDbTransaction
IDbConnection.BeginTransaction( )
```

- *IDbConnection.CreateCommand*

```
private System.Data.IDbCommand
IDbConnection.CreateCommand( )
```

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a new connection to a database with the property settings specified by the .

– **Usage**

- * The draws an open connection from the connection pool if one is available. Otherwise, it establishes a new connection to the database. Note: If a object goes out of scope, the connection it represents does not close automatically, you need to explicitly close the connection by calling or methods.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.4 CLASS FbDataAdapter

Represents a set of data commands and a connection to a data source that are used to fill the DataSet and update the data source. This class cannot be inherited. The FbDataAdapter, serves as a bridge between a DataSet and FirebirdSQL for retrieving and saving data. The FbDataAdapter provides this bridge by mapping Fill, which changes the data in the DataSet to match the data in the data source, and Update, which changes the data in the data source to match the data in the DataSet, using the appropriate DSQL statements against the data source. FbDataAdapter is used in conjunction with and to increase performance when connecting to a FirebirdSQL Server database. The FbDataAdapter also includes the , , , UpdateCommand, and TableMappings properties to facilitate the loading and updating of data. When an instance of FbDataAdapter is created, the read/write properties are set to initial values.

DECLARATION

```
public class FbDataAdapter
    : DbDataAdapter
```

PROPERTIES

- *AcceptChangesDuringFill*

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

- *Container*

```
public System.ComponentModel.IContainer Container { get; }
```

- *ContinueUpdateOnError*

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

- *DeleteCommand*

```
public FirebirdSql.Data.Firebird.FbCommand DeleteCommand {
    get; set; }
```

- *DesignMode*

```
protected bool DesignMode { get; }
```

- *Events*

```
protected System.ComponentModel.EventHandlerList Events {
    get; }
```

- *InsertCommand*

```
public FirebirdSql.Data.Firebird.FbCommand InsertCommand {
    get; set; }
```

Gets or sets an SQL statement or stored procedure used to insert new records into the data source.

- **Usage**

- * During Update, if this property is not set and primary key information is present in the DataSet, the InsertCommand can be generated automatically if you set the property and use the . Then, any additional commands that you do not set are generated by the FbCommandBuilder. This generation logic requires key column information to be present in the DataSet. When InsertCommand is assigned to a previously created , the FbCommand is not cloned. The InsertCommand maintains a reference to the previously created FbCommand object.

- *MissingMappingAction*

```
public System.Data.MissingMappingAction
MissingMappingAction { get; set; }
```

- *MissingSchemaAction*

```
public System.Data.MissingSchemaAction MissingSchemaAction
{ get; set; }
```

- *SelectCommand*

```
public FirebirdSql.Data.Firebird.FbCommand SelectCommand {
    get; set; }
```

Gets or sets an SQL statement or stored procedure used to select records in the data source.

- **Usage**

- * When SelectCommand is assigned to a previously created , the FbCommand is not cloned. The SelectCommand maintains a reference to the previously created FbCommand object. If the SelectCommand does not return any rows, no tables are added to the DataSet, and no exception is raised.

- *Site*

```
public System.ComponentModel.ISite Site { get; set; }
```

- *TableMappings*

```
public System.Data.Common.DataTableMappingCollection
TableMappings { get; }
```

- *UpdateCommand*

```
public FirebirdSql.Data.Firebird.FbCommand UpdateCommand
{ get; set; }
```

Gets or sets an SQL statement or stored procedure used to update records in the data source.

- **Usage**

- * During Update, if this property is not set and primary key information is present in the DataSet, the UpdateCommand can be generated automatically if you set the property and use the . Then, any additional commands that you do not set are generated by the FbCommandBuilder. This generation logic requires key column information to be present in the DataSet. When UpdateCommand is assigned to a previously created , the FbCommand is not cloned. The UpdateCommand maintains a reference to the previously created FbCommand object.

CONSTRUCTORS

- *.ctor*

```
public FbDataAdapter( )
```

Creates a new instance of the class.

- **Usage**

- * When you create an instance of , the following read/write properties are set to their default values, as shown in the table.

Properties	Default Value	MissingMappingAction	MissingMappingAction.Passthrough	MissingSchemaAction	MissingSchemaAction.Add

- *.ctor*

```
public FbDataAdapter( )
```

Creates a new instance of the class with the specified SQL SELECT statement.

- **Usage**

- * When you create an instance of , the following read/write properties are set to their default values, as shown in the table.

Properties	Default Value	MissingMappingAction	MissingMappingAction.Passthrough	MissingSchemaAction	MissingSchemaAction.Add

- **Parameters**

- * `selectCommand` -

- *.ctor*

```
public FbDataAdapter( )
```

- **Parameters**

- * `selectCommandText` -
- * `selectConnection` -

- *.ctor*

```
public FbDataAdapter( )
```

Creates a new instance of the class with an SQL SELECT statement and a connection string.

- **Usage**

- * When you create an instance of , the following read/write properties are set to their default values, as shown in the table.

Properties	Default Value	MissingMappingAction	MissingMappingAction.Passthrough	MissingSchemaAction	MissingSchemaAction.Add

- **Parameters**

- * `selectCommandText` -
- * `selectConnectionString` -

METHODS

• *CloneInternals*

```
protected System.Data.Common.DataAdapter CloneInternals( )
```

• *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

– **Parameters**

* requestedType -

• *CreateRowUpdatedEvent*

```
protected System.Data.Common.RowUpdatedEventArgs  
CreateRowUpdatedEvent( )
```

Creates a new instance of the RowUpdatedEventArgs class, regardless of whether the update is successful.

– **Parameters**

* dataRow -

* command -

* statementType -

* tableMapping -

• *CreateRowUpdatingEvent*

```
protected System.Data.Common.RowUpdatingEventArgs  
CreateRowUpdatingEvent( )
```

Creates a new instance of the RowUpdatingEventArgs class.

– **Parameters**

* dataRow -

* command -

* statementType -

* tableMapping -

• *CreateTableMappings*

```
protected System.Data.Common.DataTableMappingCollection  
CreateTableMappings( )
```

- *Dispose*

```
public void Dispose( )
```

- *Dispose*

```
protected void Dispose( )
```

Releases the unmanaged and, optionally, the managed resources used by the object.

– **Parameters**

* disposing -

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Fill*

```
protected int Fill( )
```

– **Parameters**

* dataTable -

* dataReader -

- *Fill*

```
protected int Fill( )
```

– **Parameters**

* dataSet -

* srcTable -

* dataReader -

* startRecord -

* maxRecords -

- *Fill*

```
protected int Fill( )
```

- **Parameters**

- * dataTable -
- * command -
- * behavior -

- *Fill*

```
protected int Fill( )
```

- **Parameters**

- * dataSet -
- * startRecord -
- * maxRecords -
- * srcTable -
- * command -
- * behavior -

- *Fill*

```
public int Fill( )
```

- **Parameters**

- * dataSet -

- *Fill*

```
public int Fill( )
```

- **Parameters**

- * dataTable -

- *Fill*

```
public int Fill( )
```

- **Parameters**

- * dataSet -
- * srcTable -

- *Fill*

```
public int Fill( )
```

– **Parameters**

- * dataSet -
- * startRecord -
- * maxRecords -
- * srcTable -

● *FillSchema*

```
protected System.Data.DataTable FillSchema( )
```

– **Parameters**

- * dataTable -
- * schemaType -
- * command -
- * behavior -

● *FillSchema*

```
protected System.Data.DataTable[] FillSchema( )
```

– **Parameters**

- * dataSet -
- * schemaType -
- * command -
- * srcTable -
- * behavior -

● *FillSchema*

```
public System.Data.DataTable[] FillSchema( )
```

– **Parameters**

- * dataSet -
- * schemaType -

● *FillSchema*

```
public System.Data.DataTable FillSchema( )
```

– **Parameters**

- * dataTable -
- * schemaType -

- *FillSchema*

```
public System.Data.DataTable[] FillSchema( )
```

– Parameters

- * dataSet -
- * schemaType -
- * srcTable -

- *Finalize*

```
protected void Finalize( )
```

- *GetFillParameters*

```
public System.Data.IDataParameter[] GetFillParameters( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetService*

```
protected object GetService( )
```

– Parameters

- * service -

- *GetType*

```
public System.Type GetType( )
```

- *ICloneable.Clone*

```
private object ICloneable.Clone( )
```

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

protected **object** MemberwiseClone()

- *OnFillError*

protected **void** OnFillError()

– **Parameters**

* value -

- *OnRowUpdated*

protected **void** OnRowUpdated()

Raises the RowUpdated event using a RowUpdatedEventArgs object.

– **Parameters**

* value -

- *OnRowUpdating*

protected **void** OnRowUpdating()

Raises the RowUpdating event using a RowUpdatingEventArgs object, whether or not the update operation is successful.

– **Parameters**

* value -

- *ShouldSerializeTableMappings*

protected **bool** ShouldSerializeTableMappings()

- *ToString*

public **string** ToString()

- *Update*

protected **int** Update()

– **Parameters**

- * dataRows -
 - * tableMapping -

- *Update*

```
public int Update( )
```

- **Parameters**

- * dataSet -

- *Update*

```
public int Update( )
```

- **Parameters**

- * dataRows -

- *Update*

```
public int Update( )
```

- **Parameters**

- * dataTable -

- *Update*

```
public int Update( )
```

- **Parameters**

- * dataSet -

- * srcTable -

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.5 CLASS FbDatabaseInfo

Reports requested information about a attached database. This class cannot be inherited.

DECLARATION

```
public class FbDatabaseInfo
: Object
```

PROPERTIES

- *ActiveTransactions*

```
public int ActiveTransactions { get; }
```

Gest the number of active transection.

- *AllocationPages*

```
public int AllocationPages { get; }
```

Gest the number of database pages allocated.

- *BackoutCount*

```
public int BackoutCount { get; }
```

Gest the number of removals of a version of a record.

- *BaseLevel*

```
public string BaseLevel { get; }
```

Gest the database version (level) number.

- *Connection*

```
public FirebirdSql.Data.Firebird.FbConnection Connection {
get; set; }
```

Gets or sets the used by this instance of the FbDataseInfo.

- *CurrentMemory*

```
public int CurrentMemory { get; }
```

Gest the Amount of server memory (in bytes) currently in use.

- *DatabaseSizeInPages*

```
public int DatabaseSizeInPages { get; }
```

Gest the database size in pages.

- *DbId*

```
public string DbId { get; }
```

Gest the database file name and site name.

- *DeleteCount*

```
public int DeleteCount { get; }
```

Gest the number of database deletes since the database was last attached.

- *ExpungeCount*

```
public int ExpungeCount { get; }
```

Gest the number of removals of a record and all of its ancestors, for records hose deletions have been committed.

- *Fetches*

```
public int Fetches { get; }
```

Gest the number of reads from the memory buffer cache.

- *ForcedWrites*

```
public bool ForcedWrites { get; }
```

Gest the mode in which database writes are performed.

- *Implementation*

```
public string Implementation { get; }
```

Gest the database implementation number.

- *InsertCount*

```
public int InsertCount { get; }
```

Gest the number of inserts into the database since the database was last attached.

- *IscVersion*

```
public string IscVersion { get; }
```

Gets the version identification string of the database implementation.

- *Marks*

```
public int Marks { get; }
```

Gest the number of writes to the memory buffer cache.

- *MaxMemory*

```
public int MaxMemory { get; }
```

Gest the Maximum amount of memory (in bytes) used at one time since the first process attached to the database.

- *NextTransaction*

```
public int NextTransaction { get; }
```

Gest the number of the next transaction.

- *NoReserve*

```
public bool NoReserve { get; }
```

Gest the indicates if space is reserved on each database page for holding.

- *NumBuffers*

```
public int NumBuffers { get; }
```

Gest the number of memory buffers currently allocated.

- *OdsMinorVersion*

```
public int OdsMinorVersion { get; }
```

Gets the On-disk structure (ODS) minor version number.

– **Usage**

- * An increase in a minor version number indicates a non-structural change, one that still allows the database to be accessed by database engines with the same major version number but possibly different minor version numbers.

- *OdsVersion*

```
public int OdsVersion { get; }
```

Gets the ODS major version number.

– **Usage**

- * ? Databases with different major version numbers have different physical layouts; a database engine can only access databases with a particular ODS major version number.
- ? Trying to attach to a database with a different ODS number results in an error.

- *OldestActiveSnapshot*

```
public int OldestActiveSnapshot { get; }
```

Gets the number of the oldest snapshot transaction.

- *OldestActiveTransaction*

```
public int OldestActiveTransaction { get; }
```

Gets the number of the oldest active transaction.

- *OldestTransaction*

```
public int OldestTransaction { get; }
```

Gets the number of the oldest transaction.

- *PageSize*

```
public int PageSize { get; }
```

Gets the page size of the database.

- *PurgeCount*

```
public int PurgeCount { get; }
```

Gest the number of removals of old versions of fully mature records.

- *ReadIdxCount*

```
public int ReadIdxCount { get; }
```

Gest the number of reads done via an index since the database was last attached.

- *ReadOnly*

```
public bool ReadOnly { get; }
```

Gets wheter database is in readonly mode.

- *Reads*

```
public int Reads { get; }
```

Gest the number of page reads.

- *ReadSeqCount*

```
public int ReadSeqCount { get; }
```

Gest the number of sequential sequential table scans (row reads) done on each table since the database was last attached.

- *ServerClass*

```
public string ServerClass { get; }
```

Gets wheter Firebird Server is a Super or Classic server.

- *ServerVersion*

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

Gets the version identification string of the Firebird Server.

- *SweepInterval*

```
public int SweepInterval { get; }
```

Gest the number of transactions that are committed between ?sweeps? to remove database record versions that are no longer needed.

- *UpdateCount*

```
public int UpdateCount { get; }
```

Gets the number of database updates since the database was last attached.

- *Writes*

```
public int Writes { get; }
```

Gets the number of page writes.

CONSTRUCTORS

- *.ctor*

```
public FbDatabaseInfo( )
```

Creates a new instance of the FbDatabaseInfo class.

- *.ctor*

```
public FbDatabaseInfo( )
```

Creates a new instance of the FbDatabaseInfo class with an object.

– **Parameters**

* connection -

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.6 CLASS FbDataReader

Provides a Way for reading a forward-only stream of rows. This class cannot be inherited. This class cannot be instantiated directly, to create a FbDataReader, you need to call the method of a object.

Until the FbDataReader is closed using the method of of the FbDataReader, the associated is busy busy serving the FbDataReader, and no other operations can be performed on the other than closing it. This is the case until the method is called. To retrieve output parameters you must close the FbDataReader. and are the only properties that you can call after the FbDataReader is closed.

DECLARATION

```
public class FbDataReader
    : MarshalByRefObject
```

PROPERTIES

- *Depth*

```
public int Depth { get; }
```

Gets the depth of nesting for the current row.

- **Usage**

- * The outermost table has a depth of zero. The Firebird .NET Data Provider does not support nesting and always returns zero.

- *FieldCount*

```
public int FieldCount { get; }
```

Gets the number of columns in the current row.

- **Usage**

- * After executing a query that does not return rows, FieldCount returns 0.

- *HasRows*

```
public bool HasRows { get; }
```

Gets whether the contains one or more rows.

- **Usage**

- * This property returns always true.

- *IsClosed*

```
public bool IsClosed { get; }
```

Gets a value indicating whether the data reader is closed.

- **Usage**

- * IsClosed and are the only properties that you can call after the is closed.

- *Item*

```
public object Item { get; }
```

Gets the value of the specified column in its native format given the column ordinal.

- **Parameters**

- * *i* -

- *Item*

```
public object Item { get; }
```

Gets the value of the specified column in its native format given the column name.

- **Parameters**

- * *name* -

- *RecordsAffected*

```
public int RecordsAffected { get; }
```

Gets the number of rows changed, inserted, or deleted by execution of the DSQL statement.

- **Usage**

- * The RecordsAffected property is not set until all rows are read and you close the .
The value of this property is cumulative. For example, if two records are inserted in batch mode, the value of RecordsAffected will be two. and RecordsAffected are the only properties that you can call after the is closed.

METHODS

- *Close*

```
public void Close( )
```

Closes the object.

- **Usage**

- * You must explicitly call the Close method when you are using the to use the associated for any other purpose.
The Close method fills in the values for output parameters, return values and RecordsAffected, increasing the amount of time it takes to close a FbDataReader.

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

- **Parameters**

- * requestedType -

- *Equals*

```
public bool Equals( )
```

- **Parameters**

- * obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetBoolean*

```
public bool GetBoolean( )
```

Gets the value of the specified column as a Boolean.

- **Usage**

- * Call IsDBNull to check for null values before calling this method.

- **Parameters**

- * *i* -

- *GetByte*

```
public byte GetByte( )
```

Gets the value of the specified column as a byte.

- **Parameters**

- * *i* -

- *GetBytes*

```
public long GetBytes( )
```

Reads a stream of bytes from the specified column offset into the buffer as an array, starting at the given buffer offset.

- **Usage**

- * If you pass a buffer that is a null reference, GetBytes returns the length of the field in bytes.

- No conversions are performed, therefore the data retrieved must already be a byte array.

- **Parameters**

- * *i* -

- * *dataIndex* -

- * *buffer* -

- * *bufferIndex* -

- * *length* -

- *GetChar*

```
public char GetChar( )
```

Gets the value of the specified column as a character.

– **Parameters**

* *i* -

• *GetChars*

```
public long GetChars( )
```

Reads a stream of characters from the specified column offset into the buffer as an array, starting at the given buffer offset.

– **Usage**

* If you pass a buffer that is a null reference, GetChars returns the length of the field in characters.

No conversions are performed, therefore the data retrieved must already be a character array.

– **Parameters**

* *i* -

* *dataIndex* -

* *buffer* -

* *bufferIndex* -

* *length* -

• *GetData*

```
public System.Data.IDataReader GetData( )
```

NOT SUPPORTED.

– **Parameters**

* *i* -

• *GetDataTypeName*

```
public string GetDataTypeName( )
```

Gets the name of the source data type.

– **Parameters**

* *i* -

• *GetDateTime*

```
public System.DateTime GetDateTime( )
```

Gets the value of the specified column as a DateTime object.

- **Parameters**

- * *i* -

- *GetDecimal*

```
public decimal GetDecimal( )
```

Gets the value of the specified column as a Decimal object.

- **Parameters**

- * *i* -

- *GetDouble*

```
public double GetDouble( )
```

Gets the value of the specified column as a double-precision floating point number.

- **Parameters**

- * *i* -

- *GetFieldType*

```
public System.Type GetFieldType( )
```

Gets the Type that is the data type of the column.

- **Parameters**

- * *i* -

- *GetFloat*

```
public float GetFloat( )
```

Gets the value of the specified column as a single-precision floating-point number.

- **Parameters**

- * *i* -

- *GetGuid*

```
public System.Guid GetGuid( )
```

NOT SUPPORTED

– Parameters

* i -

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetInt16*

```
public System.Int16 GetInt16( )
```

Gets the value of the specified column as a 16-bit signed integer.

– Parameters

* i -

- *GetInt32*

```
public int GetInt32( )
```

Gets the value of the specified column as a 32-bit signed integer.

– Parameters

* i -

- *GetInt64*

```
public long GetInt64( )
```

Gets the value of the specified column as a 64-bit signed integer.

– Parameters

* i -

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetName*

```
public string GetName( )
```

Gets the name of the specified column.

– **Parameters**

* *i* -

• *GetOrdinal*

```
public int GetOrdinal( )
```

Gets the column ordinal, given the name of the column.

– **Parameters**

* *name* -

• *GetSchemaTable*

```
public System.Data.DataTable GetSchemaTable( )
```

Returns a DataTable that describes the column metadata of the .

– **Usage**

* *DataReader* *ColumnName* *ColumnOrdinal* *ColumnSize* *NumericPrecision* *NumericScale* *ProviderType* *DataTypeMaps* *ProviderType* *IsLong* *AllowDBNull* *IsReadOnly* *IsRowVersion* *IsUnique* *IsKey*

The name of the column; this might not be unique. If this cannot be determined, a null value is returned. This name always reflects the most recent renaming of the column in the current view or command text.

The ordinal of the column. Columns are numbered starting with one. This column cannot contain a null value.

The maximum possible length of a value in the column. For columns that use a fixed-length data type, this is the size of the data type. NumericPrecision If *ProviderType* is a numeric data type, this is the maximum precision of the column. The precision depends on the definition of the column. If *ProviderType* is not a numeric data type, this is a null value. NumericScale If *ProviderType* is DECIMAL or NUMERIC data type, the number of digits to the right of the decimal point. Otherwise, this is a null value. *DataTypeMaps* to the .NET Framework type of the column. *ProviderType* The indicator of the column's data type. If the data type of the column varies from row to row, this must be *Object*. This column cannot contain a null value. *IsLong* true if the column contains a BLOB that contains very long data; otherwise false. *AllowDBNull* true if the column allows null values; otherwise false. *IsReadOnly* true if the column cannot be modified; otherwise false. *IsRowVersion* Set if the column contains a persistent row identifier that cannot be written to, and has no meaningful value except to identify the row. *IsUnique* true if the the column is one of a set of columns of a unique key. *IsKey* true if the the column is one

of a set of columns of a primary key. `IsAutoIncrementtrue` if the column assigns values to new rows in fixed increments; otherwise false. The default of this column is false. `IsAliasedTrue` if the column name is an alias; otherwise false. `IsExpressionTrue` if the column is an expression; otherwise false. `BaseSchemaName` As Firebird doesn't support Schemas this column will be always `gt;NULL`. `BaseCatalogName` As Firebird doesn't support Catalogs this column will be always `gt;NULL`. `BaseTableName` The name of the table or view in the data store that contains the column. A null value if the base table name cannot be determined. The default of this column is a null value. `BaseColumnName` The name of the column in the data store. This might be different than the column name returned in the `ColumnName` column if an alias was used. A null value if the base column name cannot be determined or if the rowset column is derived, but not identical to, a column in the data store. The default of this column is a null value.

- *GetString*

```
public string GetString( )
```

Gets the value of the specified column as a string.

- **Parameters**

- * `i` -

- *GetType*

```
public System.Type GetType( )
```

- *GetValue*

```
public object GetValue( )
```

Gets the value of the specified column in its native format.

- **Usage**

- * This method returns null value as `DBNull`.

- **Parameters**

- * `i` -

- *GetValues*

```
public int GetValues( )
```

– **Parameters**

* values -

- *IDisposable.Dispose*

```
private void IDisposable.Dispose( )
```

- *IEnumerable.GetEnumerator*

```
private System.Collections.IEnumerator
IEnumerable.GetEnumerator( )
```

This member supports the .NET Framework infrastructure and is not intended to be used directly from your code.

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *IsDBNull*

```
public bool IsDBNull( )
```

Gets a whether the column contains non-existent or missing values.

– **Parameters**

* i -

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *NextResult*

```
public bool NextResult( )
```

Advances to the next result, when reading the results of batch DSQL statements.

– **Usage**

* Used to process execution of batch DSQL statements.

By default, the data reader is positioned on the first result.

- *Read*

```
public bool Read( )
```

Advances the to the next record.

- **Usage**

- * The default position of the is prior to the first record. Therefore, you must call Read to begin accessing any data.

- Only one FbDataReader per can be open at a time, and any attempt to open another will fail until the first one is closed.

- Similarly, while the FbDataReader is in use, the associated FbConnection is busy serving it until you close the by call method.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.7 CLASS FbError

Collects information relevant to a error returned by Firebird. This class cannot be inherited. This class is instantiated by the Firebird .NET Data Provider when an error occurs. An instance of FbError is created and managed by the class.

DECLARATION

```
public class FbError
    : Object
```

PROPERTIES

- *Class*

```
public byte Class { get; }
```

Gets the severity level of the error returned from Firebird.

- *LineNumber*

```
public int LineNumber { get; }
```

Gets the line number within the DSQL command batch or stored procedure that contains the error.

– **Usage**

- * Line numbering starts at 1. If the value is 0, the line number is not applicable.

- *Message*

```
public string Message { get; }
```

Gets the text describing the error.

- *Number*

```
public int Number { get; }
```

Gets a number that identifies the type of error.

METHODS

• *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

• *Finalize*

```
protected void Finalize( )
```

• *GetHashCode*

```
public int GetHashCode( )
```

• *GetType*

```
public System.Type GetType( )
```

• *MemberwiseClone*

```
protected object MemberwiseClone( )
```

• *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.8 CLASS FbErrorCollection

Collects all errors generated by the Firebird .NET Data Provider. This class cannot be inherited. This class is created by to collect instances of the class. FbErrorCollection always contains at least one instance of the FbError class.

DECLARATION

```
public class FbErrorCollection
    : Object
```

PROPERTIES

- *Count*

```
public int Count { get; }
Gets the number of objects in the collection.
```

- *Item*

```
public FirebirdSql.Data.Firebird.FbError Item { get; set; }
Gets or sets the with the specified error Message.
```

– **Parameters**

* errorMessage -

- *Item*

```
public FirebirdSql.Data.Firebird.FbError Item { get; set; }
```

– **Parameters**

* errorIndex -

METHODS

- *CopyTo*

```
public void CopyTo( )
Copies the elements of the collection into an Array, starting at the given
index within the Array.
```

– **Parameters**

- * array -
- * index -

- *Equals*

public bool **Equals**()

– **Parameters**

- * obj -

- *Finalize*

protected void **Finalize**()

- *GetHashCode*

public int **GetHashCode**()

- *GetType*

public System.Type **GetType**()

- *IEnumerable.GetEnumerator*

private System.Collections.IEnumerator
IEnumerator.GetEnumerator()

- *MemberwiseClone*

protected object **MemberwiseClone**()

- *ToString*

public string **ToString**()

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.9 CLASS FbException

The exception that is thrown when Firebird Server returns a warning or error. This class cannot be inherited. This class is created whenever the Firebird Server .NET Data Provider encounters an error generated from the server. FbException always contains at least one instance of .

DECLARATION

```
public class FbException
    : SystemException
```

PROPERTIES

- *ErrorCode*

```
public int ErrorCode { get; }
```

Gets a value representing the Firebird error code

- *Errors*

```
public FirebirdSql.Data.Firebird.FbErrorCollection Errors { get;
}
```

Gets a collection of one or more objects that give detailed information about exceptions generated by the Firebird .NET Data Provider.

- *HelpLink*

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

- *HResult*

```
protected int HResult { get; set; }
```

- *InnerException*

```
public System.Exception InnerException { get; }
```

- *Message*

```
public string Message { get; }
```

- *Source*

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

- *StackTrace*

```
public string StackTrace { get; }
```

- *TargetSite*

```
public System.Reflection.MethodBase TargetSite { get; }
```

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetBaseException*

```
public System.Exception GetBaseException( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetObjectData*

```
public void GetObjectData( )
```

– **Parameters**

* info -

* context -

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.10 CLASS FbInfoMessageEventArgs

Provides data for the InfoMessage event. This class cannot be inherited. The InfoMessage event contains an collection with warnings messages sent from the Firebird Server.

DECLARATION

```
public class FbInfoMessageEventArgs
    : EventArgs
```

PROPERTIES

- *Errors*

```
public FirebirdSql.Data.Firebird.FbErrorCollection Errors { get; }
}
```

Gets the collection of warnings sent from the Firebird Server.

- *Message*

```
public string Message { get; }
```

Gets a value representing the complete error message sent from the Firebird Server.

METHODS

- *Equals*

```
public bool Equals( )
```

– Parameters

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.11 CLASS FbParameter

Represents a parameter of a , and optionally, its mapping to DataSet columns. This class cannot be inherited. Parameter names are not case sensitive.

DECLARATION

```
public class FbParameter
    : MarshalByRefObject
```

PROPERTIES

- *DbType*

```
public System.Data.DbType DbType { get; set; }
```

Gets or sets the DbType of the parameter.

- *Direction*

```
public System.Data.ParameterDirection Direction { get; set; }
```

Gets or sets the Direction of the parameter.

- *FbDbType*

```
public FirebirdSql.Data.Firebird.FbDbType FbDbType { get;
set; }
```

- *IsNullable*

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

Gets or sets whether the parameter accepts null values.

- *ParameterName*

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

Gets or sets the name of the .

- *Precision*

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

Gets or sets the maximum number of digits used to represent the parameter value.

- *Scale*

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

Gets or sets the number of decimal places to which parameter value is resolved.

- *Size*

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

Gets or sets the maximum size, in bytes, of the data within the column.

- *SourceColumn*

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

Gets or sets the name of the source column that is mapped to the DataSet and used for loading or returning the Value.

- *SourceVersion*

```
public System.Data.DataRowVersion SourceVersion { get; set; }
}
```

Gets or sets the DataRowVersion to use when loading value.

- *Value*

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

Gets or sets the value of the parameter.

CONSTRUCTORS

- *.ctor*

```
public FbParameter( )
```

Initializes a new instance of the FbParameter class.

- *.ctor*

```
public FbParameter( )
```

- **Parameters**

- * parameterName -
 - * value -

- *.ctor*

```
public FbParameter( )
```

- **Parameters**

- * parameterName -
 - * fbType -

- *.ctor*

```
public FbParameter( )
```

- **Parameters**

- * parameterName -
 - * fbType -
 - * size -

- *.ctor*

```
public FbParameter( )
```

- **Parameters**

- * parameterName -
 - * fbType -
 - * size -
 - * sourceColumn -

- *.ctor*

```
public FbParameter( )
```

- **Parameters**

- * parameterName -
 - * dbType -
 - * size -
 - * direction -
 - * isNullable -
 - * precision -
 - * scale -

- * sourceColumn -
- * sourceVersion -
- * value -

METHODS

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

– Parameters

- * requestedType -

- *Equals*

```
public bool Equals( )
```

– Parameters

- * obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetType*

```
public System.Type GetType( )
```

- *ICloneable.Clone*

```
private object ICloneable.Clone( )
```

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.12 CLASS FbParameterCollection

A collection of parameters associated to a . This class cannot be inherited. The number of the parameters in the collection must be equal to the number of parameter placeholders and/or named parameters in the command text, or Firebird will raise an exceptionraises an error.

DECLARATION

```
public class FbParameterCollection
: MarshalByRefObject
```

PROPERTIES

- *Count*

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

Gets the number of objects in the collection.

- *Item*

```
public FirebirdSql.Data.Firebird.FbParameter Item { get; set;
}
```

Gets or sets the with the specified name.

– Parameters

* parameterName -

- *Item*

```
public FirebirdSql.Data.Firebird.FbParameter Item { get; set;
}
```

Gets or sets the at the specified index.

– Parameters

* parameterIndex -

METHODS

• *Add*

```
public int Add( )
```

Adds the specified object to the collection.

– **Parameters**

* value -

• *Add*

```
public FirebirdSql.Data.Firebird.FbParameter Add( )
```

– **Parameters**

* param -

• *Add*

```
public FirebirdSql.Data.Firebird.FbParameter Add( )
```

Adds a to the FbParameterCollection with the parameter name and the data type.

– **Parameters**

* parameterName -

* value -

• *Add*

```
public FirebirdSql.Data.Firebird.FbParameter Add( )
```

– **Parameters**

* parameterName -

* type -

• *Add*

```
public FirebirdSql.Data.Firebird.FbParameter Add( )
```

– **Parameters**

```

    * parameterName -
    * fbType -
    * size -

```

- *Add*

```
public FirebirdSql.Data.Firebird.FbParameter Add( )
```

- **Parameters**

```

    * parameterName -
    * fbType -
    * size -
    * sourceColumn -

```

- *Clear*

```
public void Clear( )
```

Removes all items from the collection.

- *Contains*

```
public bool Contains( )
```

Indicates whether a exists in the collection.

- **Parameters**

```

    * value -

```

- *Contains*

```
public bool Contains( )
```

Indicates whether a exists in the collection.

- **Parameters**

```

    * parameterName -

```

- *CopyTo*

```
public void CopyTo( )
```

Copies FbParameterCollection to the specified array.

- **Parameters**

- * array -
- * index -

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

– Parameters

- * requestedType -

- *Equals*

```
public bool Equals( )
```

– Parameters

- * obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetEnumerator*

```
public System.Collections.IEnumerator GetEnumerator( )
```

This member supports the .NET Framework infrastructure and is not intended to be used directly from your code.

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetType*

```
public System.Type GetType( )
```

- *IndexOf*

```
public int IndexOf( )
```

Gets the location of the in the collection.

– Parameters

* value -

- *IndexOf*

```
public int IndexOf( )
```

Gets the location of the in the collection with a specific parameter name.

– Parameters

* parameterName -

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *Insert*

```
public void Insert( )
```

Gets the location of the in the collection.

– Parameters

* index -

* value -

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Remove*

```
public void Remove( )
```

Removes the specified from the collection.

– Parameters

* value -

- *RemoveAt*

```
public void RemoveAt( )
```

Removes the specified from the collection using a specific index.

- **Parameters**

- * *index* -

- *RemoveAt*

```
public void RemoveAt( )
```

Removes the specified from the collection.

- **Parameters**

- * *parameterName* -

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

1.2.13 CLASS FbRowUpdatedEventArgs

Provides data for the RowUpdated event. This class cannot be inherited.

DECLARATION

```
public class FbRowUpdatedEventArgs
    : RowUpdatedEventArgs
```

PROPERTIES

- *Command*

```
public FirebirdSql.Data.Firebird.FbCommand Command { get; }
Gets the executed when Update is called.
```

- *Errors*

```
public System.Exception Errors { get; set; }
```

- *RecordsAffected*

```
public int RecordsAffected { get; }
```

- *Row*

```
public System.Data.DataRow Row { get; }
```

- *StatementType*

```
public System.Data.StatementType StatementType { get; }
```

- *Status*

```
public System.Data.UpdateStatus Status { get; set; }
```

- *TableMapping*

```
public System.Data.Common.DataTableMapping TableMapping {
get; }
```

CONSTRUCTORS

- *.ctor*

```
public FbRowUpdatedEventArgs( )
```

Initializes a new instance of the FbRowUpdatedEventArgs class.

– **Parameters**

- * *row* -
- * *command* -
- * *statementType* -
- * *tableMapping* -

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

- * *obj* -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

1.2.14 CLASS FbRowUpdatingEventArgs

Provides data for the RowUpdating event. This class cannot be inherited.

DECLARATION

```
public class FbRowUpdatingEventArgs
    : RowUpdatingEventArgs
```

PROPERTIES

- *Command*

```
public FirebirdSql.Data.Firebird.FbCommand Command { get;
set; }
```

Gets the executed when Update is called.

- *Errors*

```
public System.Exception Errors { get; set; }
```

- *Row*

```
public System.Data.DataRow Row { get; }
```

- *StatementType*

```
public System.Data.StatementType StatementType { get; }
```

- *Status*

```
public System.Data.UpdateStatus Status { get; set; }
```

- *TableMapping*

```
public System.Data.Common.DataTableMapping TableMapping {
get; }
```

CONSTRUCTORS

- *.ctor*

```
public FbRowUpdatingEventArgs( )
```

Initializes a new instance of the FbRowUpdatingEventArgs class.

– **Parameters**

- * *row* -
- * *command* -
- * *statementType* -
- * *tableMapping* -

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

- * *obj* -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

1.2.15 CLASS FbTransaction

Represents a Firebird transaction. This class cannot be inherited.

DECLARATION

```
public class FbTransaction
    : MarshalByRefObject
```

PROPERTIES

- *Connection*

```
public FirebirdSql.Data.Firebird.FbConnection Connection {
    get; set; }
```

Gets the object associated with the transaction, or a null reference if the transaction is no longer valid.

- **Usage**

- * A single application may have multiple database connections, each with or without a transaction. This property allow you to know the connection object associated with a specific transaction created by .

- *IsolationLevel*

```
public System.Data.IsolationLevel IsolationLevel { get; set; }
```

Specifies the IsolationLevel for this transaction.

- **Usage**

- * Parallel transactions are not supported. Therefore, the IsolationLevel applies to the entire transaction.

METHODS

- *Commit*

```
public void Commit( )
```

Commits the transaction.

- *CommitRetaining*

```
public void CommitRetaining( )
```

Commits the transaction and retains the transaction context after a commit.

- *CreateObjRef*

```
public System.Runtime.Remoting.ObjRef CreateObjRef( )
```

– Parameters

* requestedType -

- *Dispose*

```
public void Dispose( )
```

Releases the unmanaged resources used by the FbTransaction and optionally releases the managed resources.

- *Equals*

```
public bool Equals( )
```

– Parameters

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetLifetimeService*

```
public object GetLifetimeService( )
```

- *GetType*

```
public System.Type GetType( )
```

- *InitializeLifetimeService*

```
public object InitializeLifetimeService( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Rollback*

```
public void Rollback( )
```

Rolls back a transaction from a pending state.

- *Rollback*

```
public void Rollback( )
```

Rolls back a transaction from a pending state, and specifies a savepoint name.

– **Parameters**

* savePointName -

– **Example**

```
FbConnection myConnection = new
FbConnection(connectionString); myConnection.Open();
FbCommand myCommand = new FbCommand(); FbTransaction
myTrans;
// Start a local transaction myTrans =
myConnection.BeginTransaction(); // Assign transaction
object for a pending local transaction myCommand.Connection
= myConnection; myCommand.Transaction = myTrans;
try { myCommand.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBNP', '.Net Provider', 'N/A')";
myCommand.ExecuteNonQuery();
myTrans.Save("SampleTransaction"); myCommand.CommandText =
"INSERT INTO PROJECT(proj_id, proj_name, product)
Values('FBN1', '.Net Provider1.', 'N/A')";
myCommand.ExecuteNonQuery(); myTrans.Commit();
Console.WriteLine("Both records are written to database.");
} catch(Exception e) { try {
myTrans.Rollback("SampleTransaction"); } catch (FbException
ex) { if (myTrans.Connection != null) {
Console.WriteLine("An exception of type " + ex.GetType() + "
```

```

was encountered while attempting to roll back the
transaction."); } }
Console.WriteLine("An exception of type " + e.GetType() + "
was encountered while inserting the data.");
Console.WriteLine("Neither record was written to
database."); } finally { myConnection.Close(); }

```

- *RollbackRetaining*

```
public void RollbackRetaining( )
```

Rolls back a transaction from a pending state and retains the transaction context after the rollback.

- *Save*

```
public void Save( )
```

Establish a new Save point for the current transaction with the given name.

- **Usage**

- * Savepoints offer a mechanism to roll back portions of transactions. You create a savepoint using the Save method, and then later call the method to roll back to the savepoint instead of rolling back to the start of the transaction.

- **Parameters**

- * savePointName -

- **Example**

```

FbConnection myConnection = new
FbConnection(connectionString); myConnection.Open();
FbCommand myCommand = new FbCommand(); FbTransaction
myTrans;
// Start a local transaction myTrans =
myConnection.BeginTransaction(); // Assign transaction
object for a pending local transaction myCommand.Connection
= myConnection; myCommand.Transaction = myTrans;
try { myCommand.CommandText = "INSERT INTO PROJECT(proj_id,
proj_name, product) Values('FBNP', '.Net Provider', 'N/A')";
myCommand.ExecuteNonQuery();
myTrans.Save("SampleTransaction"); myCommand.CommandText =
"INSERT INTO PROJECT(proj_id, proj_name, product)
Values('FBN1', '.Net Provider1.', 'N/A')";
myCommand.ExecuteNonQuery(); myTrans.Commit();
Console.WriteLine("Both records are written to database.");

```

```
    } catch(Exception e) { try {
myTrans.Rollback("SampleTransaction"); } catch (FbException
ex) { if (myTrans.Connection != null) {
Console.WriteLine("An exception of type " + ex.GetType() + "
was encountered while attempting to roll back the
transaction."); } }
Console.WriteLine("An exception of type " + e.GetType() + "
was encountered while inserting the data.");
Console.WriteLine("Neither record was written to
database."); } finally { myConnection.Close(); }
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

Chapter 2

Namespace Firebird- Sql.Data.Firebird.Events

Namespace Contents

Page

Interfaces

Classes

FbEvent??

Represents an event or a set of events that can be fired from a trigger or stored procedure. This class cannot be inherited.

FbEventAlertEventArgs??

Provides data for the EventAlert event. This class cannot be inherited.

2.1 Interfaces

2.2 Classes

2.2.1 CLASS FbEvent

Represents an event or a set of events that can be fired from a trigger or stored procedure. This class cannot be inherited.

DECLARATION

```
public class FbEvent
: Object
```

PROPERTIES

- *Connection*

```
public FirebirdSql.Data.Firebird.FbConnection Connection {
get; set; }
```

Gets or sets the used by this instance of the .

CONSTRUCTORS

- *.ctor*

```
public FbEvent( )
```

Creates a new instance of the class.

- *.ctor*

```
public FbEvent( )
```

– **Parameters**

* connection -

- *.ctor*

```
public FbEvent( )
```

– **Parameters**

- * `connection` -
- * `events` -

METHODS

- *CancelEvents*

```
public void CancelEvents( )
```

- *Equals*

```
public bool Equals( )
```

– **Parameters**

- * `obj` -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *QueEvents*

```
public void QueEvents( )
```

Request asynchronous notification of events registered with .

- *RegisterEvents*

```
public void RegisterEvents( )
```

– **Parameters**

* events -

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

2.2.2 CLASS FbEventAlertEventArgs

Provides data for the EventAlert event. This class cannot be inherited. The EventAlert event contains an array with the number of counts of each event.

DECLARATION

```
public class FbEventAlertEventArgs
    : EventArgs
```

PROPERTIES

- *Counts*

```
public System.Int32[] Counts { get; set; }
```

Returns the number of times each event has occurred.

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

protected **object** MemberwiseClone()

- *ToString*

public **string** ToString()

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

Chapter 3

Namespace

FirebirdSql.Data.Firebird.Isql

Namespace Contents

Page

Interfaces

Classes

FbScript.....??
Provides a basic parser for FirebirdSQL ISQL scripts.

3.1 Interfaces

3.2 Classes

3.2.1 CLASS FbScript

Provides a basic parser for FirebirdSQL ISQL scripts.

DECLARATION

```
public class FbScript
: Object
```

PROPERTIES

- *CommandCollection*

```
public System.Collections.ArrayList CommandCollection { get; }
```

Gets the collection of individual commands result of parse the ISQL script.

- *Script*

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

Gets or sets the ISQL script to parse.

CONSTRUCTORS

- *.ctor*

```
public FbScript( )
```

Creates a new instance of the class.

- *.ctor*

```
public FbScript( )
```

Creates a new instance of the class with the specified ISQL script.

- **Parameters**
 - * *script* -

METHODS

• *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

• *Finalize*

```
protected void Finalize( )
```

• *GetHashCode*

```
public int GetHashCode( )
```

• *GetType*

```
public System.Type GetType( )
```

• *MemberwiseClone*

```
protected object MemberwiseClone( )
```

• *Parse*

```
public void Parse( )
```

• *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

Chapter 4

Namespace Firebird-Sql.Data.Firebird.Services

Namespace Contents

Page

Interfaces

Classes

FbBackup	??
<i>Back up a database to a file or tape device. This class cannot be inherited.</i>	
FbConfiguration	??
<i>Allow configuration of local or remote databases properties.</i>	
FbLog	??
<i>Retrieve the firebird.log file from the server if the log file exists. An error is returned if the log file does not exist. This class cannot be inherited.</i>	
FbRestore	??
<i>Restore a database from backup files. This class cannot be inherited.</i>	
FbSecurity	??
<i>Allow to list, add, delete, and modify users.</i>	
FbServerProperties	??
<i>Retrieve server configuration information.</i>	
FbService	??
<i>Base class for Firebird Services implementation.</i>	
FbStatistical	??
<i>Retrieve database statistics for the database specified. This class cannot be inherited.</i>	
FbUserData	??
<i>Represents data of a user of security.fdb security database.</i>	
FbValidation	??
<i>Request a database validation. This class cannot be inherited.</i>	

4.1 Interfaces

4.2 Classes

4.2.1 CLASS FbBackup

Back up a database to a file or tape device. This class cannot be inherited. Paths of backup files are relative to the server. Since the Services Manager executes backup and restore tasks on the server host, the Services Manager reads and writes backup files on the server host.

The username and password used to connect to the services manager will be used to connect to the database for backup. This helps add some degree of security for this operation. Only the SYSDBA user or the owner of the database will be able to backup a database.

DECLARATION

```
public class FbBackup
    : FbService
```

PROPERTIES

- *BackupFiles*

```
public System.Collections.ArrayList BackupFiles { get; set; }
Gets or sets the collection of backup files.
```

- *Database*

```
public string Database { get; set; }
Gets or sets the path of the primary file of the database, from the server?s
point of view.
```

- *Factor*

```
public int Factor { get; set; }
Gets the Tape device blocking factor.
```

- *Options*

```
public FirebirdSql.Data.Firebird.Services.FbBackupFlags Options
{ get; set; }
```

Gets a value that specifies options used for database backup.

- *QueryBufferSize*

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

Gets or sets the buffer size for service query calls.

- *RoleName*

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

User Role.

- *ServerName*

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

Server name for establish the connection.

- *ServerPort*

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

Port number in the server for establish the connection.

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {  
get; }
```

Gets the current state of the service.

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

- *Verbose*

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

Gets or sets wheter the output be sent to the client.

CONSTRUCTORS

- *.ctor*

```
public FbBackup( )
```

Creates a new instance of class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

protected object MemberwiseClone()

- *Open*

public void Open()

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

protected System.Collections.ArrayList parseQueryInfo()

– Parameters

* buffer -

- *queryService*

protected System.Byte[] queryService()

Retrieves information about a firebird service based on a given set of information items.

– Parameters

* items -

- *Start*

public void Start()

Starts database backup service task.

- *startTask*

protected void startTask()

Starts the execution of a service task.

- *ToString*

public string ToString()

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

4.2.2 CLASS FbConfiguration

Allow configuration of local or remote databases properties.

DECLARATION

```
public class FbConfiguration
    : FbService
```

PROPERTIES

- *Database*

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

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

```
public int ServerPort { get; set; }
Port number in the server for establish the connection.
```

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {
get; }
Gets the current state of the service.
```

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

CONSTRUCTORS

- *.ctor*

```
public FbConfiguration( )
```

Initializes a new instance of the FbConfiguration class.

METHODS

- *ActivateShadows*

```
public void ActivateShadows( )
```

Activates a shadow file for database use.

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *DatabaseOnline*

```
public void DatabaseOnline( )
```

Bring a database back online.

- *DatabaseShutdown*

```
public void DatabaseShutdown( )
```

- Parameters

- * mode -
- * seconds -

- *Equals*

public bool Equals()

- Parameters

- * obj -

- *Finalize*

protected void Finalize()

- *GetHashCode*

public int GetHashCode()

- *GetNextLine*

public string GetNextLine()

Returns one line of output from a service task.

- *GetType*

public System.Type GetType()

- *MemberwiseClone*

protected object MemberwiseClone()

- *Open*

public void Open()

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

protected System.Collections.ArrayList parseQueryInfo()

- Parameters

* **buffer** -

- *queryService*

protected System.Byte[] queryService()

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* **items** -

- *SetAccessMode*

public void SetAccessMode()

Set the access mode of the database.

– **Parameters**

* **readOnly** -

- *SetForcedWrites*

public void SetForcedWrites()

Sets the write mode for the database.

– **Parameters**

* **forcedWrites** -

- *SetPageBuffers*

public void SetPageBuffers()

Sets the number of database page buffers.

– **Parameters**

* **pageBuffers** -

- *SetReserveSpace*

public void SetReserveSpace()

Configure the database to fill data pages when inserting new records, or reserve 20% of each page for later record deltas;

– **Parameters**

* reserveSpace -

• *SetSqlDialect*

```
public void SetSqlDialect( )
```

Sets the SQL dialect of the Firebird Database.

– **Parameters**

* sqlDialect -

• *SetSweepInterval*

```
public void SetSweepInterval( )
```

Sets the number of transactions between database sweeps.

– **Parameters**

* sweepInterval -

• *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

• *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.3 CLASS FbLog

Retrieve the firebird.log file from the server if the log file exists. An error is returned if the log file does not exist. This class cannot be inherited.

DECLARATION

```
public class FbLog
    : FbService
```

PROPERTIES

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

```
public int ServerPort { get; set; }
Port number in the server for establish the connection.
```

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {
get; }
Gets the current state of the service.
```

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

CONSTRUCTORS

- *.ctor*

```
public FbLog( )
```

Initializes a new instance of the FbLog class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

```
protected System.Collections.ArrayList parseQueryInfo( )
```

– **Parameters**

* *buffer* -

- *queryService*

```
protected System.Byte[] queryService( )
```

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *Start*

```
public void Start( )
```

Starts Firebird log service task.

- *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.4 CLASS FbRestore

Restore a database from backup files. This class cannot be inherited. The username and password used to connect to the services manager will be used to connect to the database for restore. This helps add some degree of security for this operation.

Only the SYSDBA or owner of a database may use to overwrite an existing database.

DECLARATION

```
public class FbRestore
    : FbService
```

PROPERTIES

- *BackupFiles*

```
public System.Collections.ArrayList BackupFiles { get; set; }
```

Gets or sets the collections of backup files.

- *Database*

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

Gets or sets the path of the primary file of the database, from the server's point of view.

- *Options*

```
public FirebirdSql.Data.Firebird.Services.FbRestoreFlags Options
{ get; set; }
```

Gets or sets the options used for database restore.

- *PageBuffers*

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

Gets or sets the cache size for the restored database.

- *PageSize*

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

Gets or sets the page size for the restored database.

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

```
public int ServerPort { get; set; }
Port number in the server for establish the connection.
```

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {
get; }
Gets the current state of the service.
```

- *UserName*

```
public string UserName { get; set; }
Firebird user account for login.
```

- *UserPassword*

```
public string UserPassword { get; set; }
Password for the Firebird user account.
```

- *Verbose*

```
public bool Verbose { get; set; }
Gets or sets wheter the output be sent to the client.
```

CONSTRUCTORS

- *.ctor*

```
public FbRestore( )
```

Creates a new instance of class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

`protected object MemberwiseClone()`

- *Open*

`public void Open()`

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

`protected System.Collections.ArrayList parseQueryInfo()`

– **Parameters**

* *buffer* -

- *queryService*

`protected System.Byte[] queryService()`

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *Start*

`public void Start()`

Starts Firebird database restore service task.

- *startTask*

`protected void startTask()`

Starts the execution of a service task.

- *ToString*

`public string ToString()`

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

4.2.5 CLASS FbSecurity

Allow to list, add, delete, and modify users.

DECLARATION

```
public class FbSecurity
    : FbService
```

PROPERTIES

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

```
public int ServerPort { get; set; }
Port number in the server for establish the connection.
```

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {
get; }
Gets the current state of the service.
```

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

- *UsersDbPath*

```
public string UsersDbPath { get; }
```

The path to the security database on the server; for example, /usr/interbase/.

CONSTRUCTORS

- *.ctor*

```
public FbSecurity( )
```

Initializes a new instance of the FbSecurity class.

METHODS

- *AddUser*

```
public void AddUser( )
```

– **Parameters**

* **user** -

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *DeleteUser*

```
public void DeleteUser( )
```

- **Parameters**

- * `user` -

- *DisplayUser*

```
public FirebirdSql.Data.Firebird.Services.FbUserData  
DisplayUser( )
```

- **Parameters**

- * `userName` -

- *DisplayUsers*

```
public FirebirdSql.Data.Firebird.Services.FbUserData[]  
DisplayUsers( )
```

Display information for a single user or all users in the isc4.gdb security database, respectively.

- *Equals*

```
public bool Equals( )
```

- **Parameters**

- * `obj` -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

protected **object** MemberwiseClone()

- *ModifyUser*

public **void** ModifyUser()

– **Parameters**

* *user* -

- *Open*

public **void** Open()

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

protected **System.Collections.ArrayList** parseQueryInfo()

– **Parameters**

* *buffer* -

- *queryService*

protected **System.Byte[]** queryService()

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *startTask*

protected **void** startTask()

Starts the execution of a service task.

- *ToString*

public **string** ToString()

EXTENDED INFORMATION

- Assembly: `FirebirdSql.Data.Firebird`

4.2.6 CLASS FbServerProperties

Retrieve server configuration information.

DECLARATION

```
public class FbServerProperties
    : FbService
```

PROPERTIES

- *DatabasesInfo*

```
public FirebirdSql.Data.Firebird.Services.FbDatabasesInfo
    DatabasesInfo { get; }
```

Gets information about databases on server.

- *Implementation*

```
public string Implementation { get; }
```

Gets the implementation of the Firebird server.

- *LockManager*

```
public string LockManager { get; }
```

Gets the location of the InterBase lock manager file on the server; this is the value of the \$INTERBASE_LCK system environment variable, or by default \$INTERBASE/serverhostname.lck.

- *MessageFile*

```
public string MessageFile { get; }
```

Gets the location of the InterBase message file on the server; this is the value of the \$INTERBASE_MSG system environment variable, or by default \$INTERBASE/interbase.msg.

- *QueryBufferSize*

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

Gets or sets the buffer size for service query calls.

- *RoleName*

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

User Role.

- *RootDirectory*

```
public string RootDirectory { get; }
```

Gets the location of the Firebird root directory on the server; this is the value of the \$INTERBASE system environment variable, or the contents of the registry key.

- *ServerConfig*

```
public FirebirdSql.Data.Firebird.Services.FbServerConfig  
ServerConfig { get; }
```

Gets the parameters and values for IB_CONFIG.

- *ServerName*

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

Server name for establish the connection.

- *ServerPort*

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

Port number in the server for establish the connection.

- *ServerVersion*

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

Gets the version of the Firebird server.

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {  
get; }
```

Gets the current state of the service.

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

- *Version*

```
public int Version { get; }
```

Gets the version of the services manager.

CONSTRUCTORS

- *.ctor*

```
public FbServerProperties( )
```

Initializes a new instance of the FbServerProperties class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

```
protected System.Collections.ArrayList parseQueryInfo( )
```

– **Parameters**

* *buffer* -

- *queryService*

```
protected System.Byte[] queryService( )
```

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.7 CLASS FbService

Base class for Firebird Services implementation.

DECLARATION

```
public class FbService
    : Object
```

PROPERTIES

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

```
public int ServerPort { get; set; }
Port number in the server for establish the connection.
```

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {
get; }
Gets the current state of the service.
```

– Usage

- * The allowed state changes are: From Closed to Open, using the Open method of the service object.
From Open to Closed, using either the Close method or the Dispose method of the service object.

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

CONSTRUCTORS

- *.ctor*

```
protected FbService( )
```

Creates a new instance of the class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

```
protected System.Collections.ArrayList parseQueryInfo( )
```

– **Parameters**

* *buffer* -

- *queryService*

```
protected System.Byte[] queryService( )
```

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.8 CLASS FbStatistical

Retrieve database statistics for the database specified. This class cannot be inherited.

DECLARATION

```
public class FbStatistical
: FbService
```

PROPERTIES

- *Database*

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

Path of the primary file of the database, from the server's point of view.

- *Options*

```
public FirebirdSql.Data.Firebird.Services.FbStatisticalFlags
Options { get; set; }
```

Gets a value that specifies options used for retrieve database statistics.

- *QueryBufferSize*

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

Gets or sets the buffer size for service query calls.

- *RoleName*

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

User Role.

- *ServerName*

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

Server name for establish the connection.

- *ServerPort*

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

Port number in the server for establish the connection.

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {  
    get; }  
}
```

Gets the current state of the service.

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

CONSTRUCTORS

- *.ctor*

```
public FbStatistical( )  
}
```

Initializes a new instance of the FbStatistical class.

METHODS

- *Close*

```
public void Close( )  
}
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )  
}
```

– Parameters

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

```
protected System.Collections.ArrayList parseQueryInfo( )
```

– **Parameters**

* *buffer* -

- *queryService*

```
protected System.Byte[] queryService( )
```

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* *items* -

- *Start*

```
public void Start( )
```

- *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.9 CLASS FbUserData

Represents data of a user of security.fdb security database.

DECLARATION

```
public class FbUserData
    : Object
```

PROPERTIES

- *FirstName*

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

First name of person using this user name.

- *GroupID*

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

GroupID number, defined in /etc/group, to assign to the user.

- *GroupName*

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

Group name, as defined in /etc/group, to assign to the user in isc4.gdb reserved for future implementation.

- *LastName*

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

Last name of person using this user name.

- *MiddleName*

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

Middle name of person using this user name.

- *RoleName*

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

Optional SQL role to adopt when administering users (reserved for future use).

- *UserID*

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

User ID number, defined in `/etc/passwd`, to assign to the user.

- *UserName*

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

User name.

- *UserPassword*

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

Password for the user.

CONSTRUCTORS

- *.ctor*

```
public FbUserData( )
```

Initializes a new instance of the `FbUserData` class.

METHODS

- *Equals*

```
public bool Equals( )
```

– **Parameters**

* `obj` -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird

4.2.10 CLASS FbValidation

Request a database validation. This class cannot be inherited. Database validation scans internal data structures for specific types of corruption. In some cases, the validation operation can repair corruption.

The validation operation cannot guarantee to repair all cases of corruption. Do not rely on database validation as a disaster recovery policy in lieu of making regular backups of your database.

DECLARATION

```
public class FbValidation
    : FbService
```

PROPERTIES

- *Database*

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

- *Options*

```
public FirebirdSql.Data.Firebird.Services.FbValidationFlags
Options { get; set; }
```

- *QueryBufferSize*

```
public int QueryBufferSize { get; set; }
Gets or sets the buffer size for service query calls.
```

- *RoleName*

```
public string RoleName { get; set; }
User Role.
```

- *ServerName*

```
public string ServerName { get; set; }
Server name for establish the connection.
```

- *ServerPort*

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

Port number in the server for establish the connection.

- *State*

```
public FirebirdSql.Data.Firebird.Services.FbServiceState State {  
    get; }
```

Gets the current state of the service.

- *UserName*

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

Firebird user account for login.

- *UserPassword*

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

Password for the Firebird user account.

CONSTRUCTORS

- *.ctor*

```
public FbValidation( )
```

Creates a new instance of class.

METHODS

- *Close*

```
public void Close( )
```

Closes the connection to the Firebird Service Manager. This is the preferred method of closing any open connection.

- *Equals*

```
public bool Equals( )
```

– Parameters

* obj -

- *Finalize*

```
protected void Finalize( )
```

- *GetHashCode*

```
public int GetHashCode( )
```

- *GetNextLine*

```
public string GetNextLine( )
```

Returns one line of output from a service task.

- *GetType*

```
public System.Type GetType( )
```

- *MemberwiseClone*

```
protected object MemberwiseClone( )
```

- *Open*

```
public void Open( )
```

Opens a connection to the Firebird Services Manager.

- *parseQueryInfo*

```
protected System.Collections.ArrayList parseQueryInfo( )
```

– Parameters

* buffer -

- *queryService*

```
protected System.Byte[] queryService( )
```

Retrieves information about a firebird service based on a given set of information items.

– **Parameters**

* items -

- *Start*

```
public void Start( )
```

- *startTask*

```
protected void startTask( )
```

Starts the execution of a service task.

- *ToString*

```
public string ToString( )
```

EXTENDED INFORMATION

- Assembly: FirebirdSql.Data.Firebird