DirectODBC v1.60
Delphi ODBC Components

by

Guernsey Software Company
Dan Daley
Mariah Hinkley

The Components

Connection Component

Published properties
AutoCommit
type: boolean
values: true/false
Description
AutoCommit determines whether or not a commit is generated automatically for you after each statement. Set this to true if you want every statement committed. When set to false, you are responsible for issuing commits/rollbacks.

Setting AutoCommit to false can speed processing of multiple statements (such as during a call to update on the query component). Also note that if AutoCommit is set to true, you will not be able to rollback statements during an update if a failure occurs. For example, if you call update on a result set which has had 10 records changed, and the fifth one fails, you will not be able to rollback the first four updates when AutoCommit is true.

Connected
type: boolean
values: true/false
Description
Connected controls the first connection slot. Setting connected to true causes the connection component to connect on the first connection channel. This is provided for backward compatiblity and for design time connection.

The connection component can support up to 12 connections at any one time. You can connect on a particular connection at run time by assigning to the public property IConnected[i] where i is the channel you want connected. So Connected is equivalent to IConnected[1]. You can determine which connection a query is issued on (as well as any updates associated with that query) by setting the ConnectionNo property on the query component to a value between 1 and 12 (see query component).

ConnectionMethod
type: TConnMethod
values: cmConnect/cmDriverConnect
Description
Connection method determines how information is passes to create a connection. Setting ConnectionMethod to cmConnect causes the connection component to only use the UserID, Password, and DataSource to create a connection.

If additional information is required to establish a connection then you should set this to cmDriverConnect and use the ConnectionStr to pass the necessary information. When enough information is provided in the ConnectionStr, a connection is established. If the information is not sufficient to create the connection then the driver will prompt for additional information. This information will be appending to the supplied ConnectionStr.

Other connection methods may be provided in the future.

ConnectionStr

type: String
values: driver dependent
Description
The ConnectionStr property is used to pass driver specific information to a particular driver. See documentation on the specific driver for more information.

DataSource
type: String
values: an existing datasource (ODBC datasource or DB2 cataloqued database).
Description
The DataSource property specifies the datasource to which you want to establish a connection. This property is required when using the cmConnect method and you want to connect to a datasource other than the default datasource.

IsolationLvl
type: TIsolation
values: tiDefault/tiUnCommitted/tiCommitted/tiRepeatable/tiSerializable/tiVersioning
Description
IsolationLvl determines the type of transaction isolation you want to use. Possible values will depend on the server you are using. See the documention for you server for more information on isolation levels.

Password
type: String
values: any valid string
Description
Password is used when cmConnect method is used. This is passed as the password by which to establish a connection to the datasource.

Procedures
Description
Procedures is used to get information about stored procedures. For datasources that support this, it will provide a list of stored procedures and allow you to get information about the columns/parameters of the stored proc.

Tables
Description
Tables is used to get information about tables in a database. For datasource that support this, it will provide a list of tables and allow you to get information about the fields/columns in the table. This is usefull when trying to determine the values to set for the parameters in a parameterized query.

UserID
type: String
values: any valid string
Description
UserID is used when cmConnect method is used. This is passed as the user id by which to establish a connection to the datasource.

Events
TCommitEvent = procedure(Sender: TComponent) of Object;
TConnectEvent = procedure(Sender: TComponent) of Object;

BeforeCommit: TCommitEvent
BeforeCommit is called when the Commit method is invoked. It is called prior to the commit being issued to ODBC. This event can be used to prevent the commit from being issued by raising an exception.

BeforeConnect: TConnectEvent
BeforeConnect is called when a connection is attempted, but prior to calling the ODBC API connect routines. This provides the opportunity to prompt the user for connection information or other pre-connection initialization.

BeforeDisconnect: TConnectEvent
This event is called when a disconnect is requested, but prior to the disconnect being issued.

BeforeRollback: TCommitEvent
BeforeRollback is called when the Rollback method is invoked. It is called prior to the rollback being issued to ODBC. This event can be used to prevent the rollback from being issued by raising an exception.

Public Procedures
Commit
Defined as ICommit(1). Provided for backward compatibility.

ICommit(cIndex: TConnection)
ICommit issues a commit on connection number *cIndex*. cIndex is an integer value between 1 and 12.

CommitAll
CommitAll will issue a commit on all connections. This is not isolated to a particular datasource. It will issue a commit on all connections for all datasources.

Rollback
Defined as IRollback(1). Provided for backward compatibility.

IRollback(cIndex: TConnection)
IRollback issues a rollback on connection number *cIndex*. cIndex is an integer value between 1 and 12.

RollbackAll
RollbackAll will issue a rollback on all connections. This is not isolated to a particular datasource. It will issue a rollback on all connections for all datasources.

SetActiveState(AQuery: TGQuery; Active: Boolean)
Internal use.

SetPreparedState(AQuery: TGQuery; Prepared: Boolean)
Internal use.

Public Functions
AnyConnection
Return Value Type: Boolean
Returns true if any connection channel is active.

GetActiveCount
Return Value Type: LongInt
Returns the number of active queries using this connection component.

GetPreparedCount
Return Value Type: LongInt
Returns the number of prepared queries using this connection component.

Public Properties
Connections
Type: TConnectionSet
Values: set of numbers from 1 to 12
Description
Read Only. Connections is a set of values from 1 to 12 that includes all of the channels which have active connections. For example if channels 1 and 3 are active then Connections will be [1, 3].

DBC[cIndex: TConnection]
Type: pointer
Description
ReadOnly. DBC contains an array of connection handles. This is mainly for internal use.

IConnected[I: TConnection]
Type: Boolean
values: true/false
Description
IConnected is used to toggle a connection for any given channel. To connect on channel 3 issue IConnected[3] := True.

Query Component

Published properties
Active
Type: boolean
values: true/false
Description
Active opens and closes the cursor. Set this property to true to activate a statement which returns a result set.

Connection
Type: String
Values: any string value which corresponds to the datasource property of an available connection component.
Description
Connection determines which datasource to use to activate the statement. The drop down list will show all datasources for which there is an available connection component.

ConnectionNo
Type: TConnection
Values: integer from 1 to 12
Description
Connection connection component can maintain up to 12 connections (may vary depending on the server and the driver). The ConnectionNo property allows you to specify which connection to issue this query. This is also the connection that will be used to issue updates in the case of an editable result set.

EditFields
Type: TStrings
Description
EditFields is a list of fields contained in UpdateTable (see below) that you want updated in your next call to Update. This property can be edited directly by assigned to the list or indirectly by using the UpdateTable property editor.

KeyFields
Type: TStrings
Description
KeyFields is a list of fields contained in UpdateTable (see below) that you want to use as key fields to uniquely identify a row in your next call to Update. This property can be edited directly by assigned to the list or indirectly by using the UpdateTable property editor. Any non-blob type fields can be listed in this property.

Params
Type: TGParams
Description
The Params property contains a list of parameter objects (TGParam) for the current statement. The parameters are re-created each time the statement is set. The properties for the parameter objects must be set prior to executing the statement (though, can be done after the statement has been prepared).

RetrieveAsNeeded

Type: Boolean
Values: true/false
Description
Use RetrieveAsNeeded to inform the query component to only download records as they are requested (via a call to Next, Last, or RecordNo). A call to Last will force the entire result set to downloaded.

SQL
Type: TStrings
Description
SQL contains the statement that you want to execute. You can use the ODBC syntax for maximum portability (see an ODBC reference) or server specific syntax.

UpdateTable
Type: String
Values: any valid table name
Description
UpdateTable should contain the name of the table to issue updates against. All fields listed in EditFields and KeyFields must be in this table. By changing these three properties between calls to Update, you can update queries that involve more than one table.

Events
TScrollType = (stFirst, stLast, stNext, stPrior, stAbsolute);
TScrollEvent = procedure(Sender: TComponent; ScrollBy: TScrollType; ScrollInfo: LongInt) of Object;
TQueryActivateEvent = procedure(Sender: TComponent) of Object;
TUpdateEvent = procedure(Sender: TComponent) of Object;
TEditEvent = procedure(Sender: TComponent) of Object;
TOnFieldTextEvent = procedure(AField: TGField; var Value: String) of Object;
TRowChangedEvent = procedure(Sender: TComponent) of Object;
TRowsAffectedEvent = procedure(Sender: TComponent; Operation: TOperationTypes; RowsAffected: LongInt) of Object;
TUpdateErrorEvent = procedure(Sender: TComponent; Stmt: pointer; Operation: TOperationTypes; Response: TModalResult) of Object;

After/BeforeActivate: TQueryActivateEvent
Occur before and after the query is activated. That is when Active is set to true, these events are triggered.

After/BeforeClose: TQueryActivateEvent
Occur when Active is set to false.

After/BeforeDeleteRec: TEditEvent
Occur when a record is deleted.

After/BeforeEditRec: TEditEvent
Occur before a record is edited for the first time since Save was last called.

After/BeforeFirst: TScrollEvent
Occur when First is called.

After/BeforeInsertRec: TEditEvent
Occur when a record is inserted. You can use the after insert event to initialize new records.

After/BeforeLast: TScrollEvent
Occur when Last is called.

After/BeforeNext: TScrollEvent
Occur when Next is called.

After/BeforePrior: TScrollEvent
Occur when Prior is called.

After/BeforeSave: TEditEvent
Occur when the current record buffer is going to be saved to the list of record buffers.

After/BeforeScroll: TScrollEvent
Occur when any scrolling takes place.

After/BeforeUpdate: TUpdateEvent
Occurs when Update is called.

OnGet/OnSetFieldText: TOnFieldTextEvent
See Formatting display data in Users Guide.

RowChanged: TRowChangedEvent
This event is called whenever a new record buffer is made active. This can happen when scrolling, saving, undo-ing, deleting, cancelling edits, inserting, etc. This event is called whenever all controls should refresh their data.

RowsAffected: TRowsAffectedEvent
See Handling problems during the update operation in Users Guide.

UpdateError: TUpdateErrorEvent
See Handling problems during the update operation in Users Guide.

Public Procedures
Append
Call this method to append an empty record to the end of the record buffers

Cancel
Cancel will cancel edits to the current record.

Delete
Delete the current record

Execute
Use execute to send non-select statements to the DBMS, such as inserts/updates/deletes or DDL.

Insert
Insert an empty record buffer at the current position in the result set

ResetMode
Internal use

ResetRowFlags
This method clears the "dirty" flags for the current record.

ResetRows
ResetRows clears the "dirty" flags for all rows in the result set. This should be called after a successful series of calls to Update. That is, once all of the current updates have been sent to the server, this should be called to reset all of the flags and clear out the deleted record buffers.

Save
Save any changes to the current record into the list of record buffers

SaveMode
Internal use

Sort(SortFields: string)
Call sort to re-sort the current result set. SortFields is a string contains a comma separated list of field names on which to do the sort. The default sort order is ascending. To perform a descending sort on a field add a *des* after the field name. For example to sort on a department number field ascending and employee last name descending: SortFields = "deptno, lastname des, firstname des". Sort order is not maintained as records are added and edited. Sort must be called after any edits that would cause the order to become unsorted.

UnDelete
Call UnDelete when positioned on a deleted record to restore it to the result set. That is the record will be removed from the list of deleted records and replaced in the result set buffers.

UnDo
Call UnDo when positioned on a modifed record to return the record to its original state. The original state will be the state the record was in when it was first fetched from the server

Update
Update will send all edits (updates/inserts/deletes) to the server for fields currently in the EditFields list. To update multiple tables, reassign values to the EditFields, KeyFields, and UpdateTable properties between calls to update.

NotifyControls(NotifyEvent: TLinkEvent; Info: LongInt)
Internal use

Public Functions
BeforeEdit
Return Value: Boolean
Internal use. BeforeEdit is call by the field objects before a value is assigned to them. If BeforeEdit returns false, the field rejects the assignment.

FieldByName(FieldName: String)
Return Value: TGField
Attempts to locate a field in the result set based on the field name. If the field is not found, the an exception is raised. If the field is found, then the field object is returned.

Find(FindVals: array of const)
Return Value: boolean
Attempts to find a record based on the field values passed in FindVals. The function assumes that the result set is sorted on the fields you are attempting to search on. The values are compared from left to right with the fields listed in the call to sort in order to find a match. The number of values passed to Find must be equal to or less than the number of fields listed in the call to Sort.

To find the first employee with a last name starting with the letter O in dept number 30 (using the sort mentioned above for the sort procedure) call Find([30, 'O']);

Find returns true if an exact match is found. Otherwise it returns false. The record position will be as follows:
the next largest record if there is a record with a value larger than the search value, otherwise will be positioned on the last record of the result set.

FindField(FieldName: String)
Return Value: TGField
Attempts to locate a field in the result set based on the field name. If the field is not found, nil is returned. If the field is found, then the field object is returned.

First

Positions on the first record in the result set

Last
Makes the last record of the result set then active record. If RetrieveAsNeeded is true, then calling last will download the remaining records of the result set.

Next
Makes the record following the current record active

Prior
Makes the record preceding the current record active

Public Properties
BOF
Type: Boolean
Read Only. BOF is true if Prior is called when positioned on the first record.

Buffer
Type: TBuffers
Values: bData/bOriginal/bDeleted
Description
Buffer determines the record buffers which are currently active. bData is the default value and makes the editable record buffers active. Setting Buffer to bOriginal will make the original version of any edited records visible. bDeleted will make any deleted records visible. By making the deleted records visible, you can UnDelete records.

BufferSize
Type: LongInt
Description
Read Only. BufferSize is used internally and represents the amount of memory required for each record buffer. This includes memory for row and field status flags as well as field data. Only four bytes is allocated for blob data in order to store a handle to a global memory object.

ControlsDisabled
Type: Boolean
Description
Mainly used internally. Set this to True when performing operations which cause extensive scrolling in a query object. Internally this is set when Update is called to avoid redrawing of data controls as the active buffer is scrolled through the result set.

CursorEOF
Type: Boolean
Description
Read Only. CursorEOF is set to true when the underlying cursor has reached the end of the result set. As soon as the cursor reaches the end of the result set, the cursor is closed.

EOF
Type: Boolean
Description
Read Only. EOF is true if Next is called when positioned on the last record.

ExtendedInfo
Type: TGExtFields
Description
ExtendedInfo is mainly used internally. It is used to stream extended field information to and from disk.

Field[I: Integer]

Type: TGField
Description
Returns the I'th field of the record. This is a 1 based array of field objects.

FieldCount
Type: LongInt
Description
Returns the total number of fields in the result set.

Prepared
Type: Boolean
Description
Use prepared to improve performance when a query will be executed multiple times. Be sure to set prepared to false when the query will no longer be needed to free up resources.

QueryMode
Type: TQueryMode
Values: qmViewing/qmEditing/qmInserting/qmClosed
Description
QueryMode returns the status of the query component.

RecordCount
Type: LongInt
Description
RecordCount returns the number of record buffers in the active buffers. If RetrieveAsNeeded is true, then this number may not reflect the total number of records in the result set if all of the records have not been retrieved. This number will reflect the number of records that have been actually retrieved.

RecordNo
Type: LongInt
Description
RecordNo is a number between 1 and RecordCount which represents the active record.

RecordSize
Type: LongInt
Description
RecordSize is for internal use only. It represents the number of bytes of storage required for the field data in a record. Note that blobs only occupy four bytes of storage in the record buffer to store a handle to a global memory object.

Params Object

Public Procedures
AddParam(AParam: TGParam)
internal use.
RemoveParam(AParam: TGParam)
internal use.

Public Functions
Count
Return Value: LongInt
Returns the number of parameters in the sql statement

Public Properties
Param[Index: LongInt]
Type: TGParam
Returns the I'th parameter object

Param Object

Public Procedures
BindParam(DBC: pointer; Stmt: pointer)
Internal use.
Public Properties

AsString
Type: String
Read or write the parameter value as a string

AsInteger
Type: LongInt
Read or write the parameter value as an integer

AsFloat
Type: Double
Read or write the parameter value as a double

AsDate
Type: Date_Struct
Read or write the parameter value as a date

AsTime
Type: Time_Struct
Read or write the parameter value as time

AsTimeStamp
Type: TimeStamp_Struct
Read or write the parameter value as time stamp/date time

BufferSize
Type: LongInt
Internal use

DataLen
Type: LongInt
Meaning depends upon the SQLType. This corresponds to the precision value in ODBC and DB2. For character values this is the length of the string. For decimal/numeric types this is the number of digits in the number.

ParamName
Type: String
This is the name of the parameter in the statement. This is the name to use to access the parameter when using ParamByName to access the parameter.

Scale
Type: Integer
Meaning depends upon the SQLType. This corresponds to the scale value in ODBC and DB2. For decimal/numeric types this is the number of digits after the decimal place in the number.

ParamType
Type: Integer
Values: SQL_PARAM_INPUT/SQL_PARAM_INPUT_OUTPUT/SQL_PARAM_RESULT_COL/SQL_PARAM_OUTPUT/
SQL_PARAM_RETURN_VALUE
Most common values for this property are input, input_output, or output. An input parameter is a parameter that does not

recieve a value from the statement (such as parameters in the where clause of a select statement). An input_output parameter is a parameter that is to pass values to the server as well as recieve values back from the server (as in parameters to stored procedures that are input/output parameters). An output parameter is a parameter that is to recieve values back from the server (as in parameters to stored procedures that are output parameters).

SQLType
Type: Integer
Values:
SQL_Char, SQL_Numeric, SQL_Decimal, SQL_Integer, SQL_SmallInt, SQL_Float, SQL_Real, SQL_Double, SQL_VarChar,
SQL_Date, SQL_Time, SQL_TimeStamp, SQL_LongVarChar, SQL_Binary, SQL_VarBinary, SQL_LongVarBinary, SQL_BigInt
SQL_TinyInt, SQL_Bit
This property informs the driver as to the underlying field type for the parameter. This value should be the same as the field to which the parameter is being compared (use the tables property of the connection component to determine the field type).

IsNull
Type: boolean
Values: True/False
Set IsNull to true to pass a null value in a parameter to the server. If IsNull is false, then the value assinged via one of the assignment properties mentioned above is used.

Field Object

Public Procedures
BindColumn(RecBuffer: PChar; RecSize: LongInt)
Internal use

Public Properties
AsDate
Type: Date_Struct
Description
Returns the field value in a date structure. Date and Timestamp field types can be access via the AsDate property.

AsFloat
Type: Double
Description
Returns the field value as a double. Float field types can be accessed using AsFloat.

AsInteger
Type: LongInt
Description
Returns an integer value for the field. All integer types can be access using AsInteger.

AsString
Type: String
Description
AsString returns a string representation of the field value. All non-blob fields can be accessed as string.

AsText
Type: String
Description
AsText returns a string representation of the field value after calling OnGetText or OnSetText of the quer component. All non-blob fields can be accessed as text.

AsTime
Type: Time_Struct
Description
Returns the field value in a time structure. Time and Timestamp field types can be access via the AsTime property.

AsTimeStamp
Type: TimeStamp_Struct
Description
Returns the field value in a timestamp structure. Timestamp field types can be access via the AsTimeStamp property.

BufferSize
Type: LongInt
Description
BufferSize is the number of bytes that this field occupies in the record buffer. Blob fields, for instance, have a BufferSize of 4, which is the size of THandle.

DataLen
Type: LongInt
Description
Read Only. DataLen is the length of the field as returned by the driver.

DataType
Type: Integer
Values: SQL_C_CHAR/SQL_C_LONG/SQL_C_SHORT/SQL_C_FLOAT/SQL_C_DOUBLE
SQL_C_DATE/SQL_C_TIME/SQL_C_TIMESTAMP/SQL_C_BINARY/SQL_C_BIT/SQL_C_TINYINT
Description
Read Only. DataType is the format that the field reads the data from the driver.

DisplayLabel
Type: String
Description
DisplayLabel is used by the grid to determine the column heading for this field.

DisplaySize
Type: LongInt
Description
DisplaySize is used by the grid to determine the initial column width for this field.

FieldName
Type: String
Description
The name of the field as returned by the driver. Note that some drivers/DBMSs may rename duplicate columns (see TableFieldName property to be able to correctly update renamed columns).

FieldNo
Type: LongInt
Description
Read Only. Internal use.

IsNull
Type: Boolean
Description
IsNull is a read/write property to set the null status of a field. Set IsNull to true to set the field to null.

Nullable
Type: Integer

Values: SQL_NO_NULLS/SQL_NULLABLE/SQL_NULLABLE_UNKNOWN
Description
ReadOnly. Use this property to determine whether or not this field can accept null values.

Offset
Type: LongInt
Description
Internal use. This is the offset into the record buffer at which this fields data is stored.

Precision
Type: LongInt
Description
See DataLen property for the Param Object

Scale
Type: Integer
Description
See Scale for the Param Object

SQLType
Type: Integer
Description
See SQLType for the Param Object

SortDir
Type: TSortDir
Values: sdAsc/sdDesc
Description
If this field is currently used for sorting then SortDir determines whether the order is ascending (sdAsc) or descending for this field. See Sort(...) method of the Query component.

Status
Type: TFieldStat
Values: fsOriginal/fsModified
Description
Read Only. An indicator to the status of this field for the current record. Returns fsModified if this field has been edited since the record has been retrieved or since the last call to ResetRows.

TableFieldName
Type: String
Description
Table Field Name is the name of the actual underlying field in the database. Usually this is the same as the FieldName, however in the case of aliased fields the two properties could differ. Use the UpdateTable property editor of the Query component to edit this value at design time.

TableName
Type: String read GetTableName;
Description
Not used currently.

Value
Type: PChar
Description
Read Only. Returns the raw data buffer as retrieved from the driver. Mainly for internal use.

Binary/Blob Field Objects

Public Procedures
LoadFromFile(FileName: String)
LoadFromFile reads the specified file into the blob field.

LoadFromStream(aStream: TStream)
LoadFromStream will read data from any stream into the blob field.

ReadBlock(Offset, Size: LongInt; Buffer: PChar; var BytesRead: LongInt)
ReadBlock will read a portion of the blob into the provided buffer. The memory for the buffer must have been allocated prior to the call to ReadBlock. BytesRead is set to the actual number of bytes copied into the provided buffer. Offset is the position in the blob to begin reading data (0 is the first byte). Size is the number of bytes to read. Size is *not* limited to 64k.

SaveToFile(FileName: String)
SaveToFile will write the blob data to the specified file. If a file already exists with the provided name, it will be overwritten.

SaveToStream(aStream: TStream)
SaveToStream will write the blob data to any stream. See Delphi help for more information on streams.

Truncate(NewSize: LongInt)
Use Truncate to reduce the size of a blob. This is only necessary if you use write block to write a new blob value to the field that is smaller than the current blob.

WriteBlock(Offset, Size: LongInt; Buffer: PChar)
Use WriteBlock to write raw binary data to the field. Offset is the starting position in the blob to begin writting data (0 is the first byte). If Offset + Size is larger than the current blob, new memory will be allocated automatically. However, Offset must be a value within the current blob.

Binary Streams
See Delphi On-Line help for information on streams. These components provide a stream object that is a standard stream. The only difference is the constructor. The constructor takes a reference to a binary field object. For example, to create a binary stream for an ODBC binary field:

```
MyStream := TODBCBinaryStream.Create(ODBCQuery1.FieldByName('MyBlobField'));
```

Extended Field Information Objects
Extended Field Information Objects are used internally to stream field information to and from disk. They are used to store information about the field such as DisplayLabel and TableFieldName. These provide the mechanism to allow you to specify extended field information at design time and save it with the form.

Public Properties
DisplayLabel
Type: String
Description
See DisplayLabel property for the Field Objects

DisplayMask
Type: String
Description
Not implemented yet.

FieldName
Type: String
Description
Name of the associated field

TableFieldName
Type: String
Description
See TableFieldName property for the Field Objects

Extended Fields Information Objects

Extended Fields Information Objects are used to store a list of Extended Field Information Objects. They are used internally by the Query component.

Public Procedures
AddExtInfo(AExtInfo: TGExtFieldInfo)
Description
Adds an extended field information object to the current list.

Refresh
Refresh resynchronizes the extended fields information list with the columns returned by the current statement. If a field for a corresponding extended field information object is no longer returned by the statement, then the object is remove from the list. If fields are returned which do not have extended information objects, then objects are created and added to the current list.

RemoveExtInfo(AExtInfo: TGExtFieldInfo)
Remove the extended field information object from the list.

Public Functions
Count
Return Type: LongInt;
Description
Returns the number of extended field information objects in the current list.

ExtInfoForField(aFieldName: String)
Return Type: TGExtFieldInfo
Description
Returns the extended field information object for the provided field name.

Public Properties
ExtInfo[Index: LongInt]
Type: TGExtFieldInfo
Description
Read Only. Returns the extended field information object that is at Index in the list.