Remote Data Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"}

RDO provides built-in constants that you can use with methods, properties and events. These constants all begin with the letters "rd" and are documented with the event, method or property to which they apply. These constants are also used with **RemoteData** Control properties.

You can use the Object Browser to browse the list of built-in constants. From the View menu, choose Object Browser and select the appropriate library. Scroll the list in the Classes portion of the dialog to see the constants groups. To find a specific constant, use the search window to locate a specific group or constant.

Remote Data Constants	Description – Determines:
Attributes Property	**rdoColumn** object characteristics
BOFAction Property	Beginning of file options
CursorDriver Property	Type of cursor library
Column Status	Batch mode update column status
Data Type	Data type
Direction	Parameter type
EditMode	**Edit** or **AddNew** state
EOFAction	End of file options
Error Event	Error message handling
LockType	Type of concurrency management.
Option	Processing options
Prompt	ODBC driver manager prompting
QueryType	Type of query
rdoDefaultCursorDriver Property	Type of cursor library
rdoLocaleID	Error message language
ResultsetType	Type of cursor
Row Status	Batch mode update row status
SQLRetCode	ODBC return codes
Status (rdoColumn)	Batch mode update column status
Status (rdoResultset)	Batch mode update row status
Type (rdoColumn, rdoParameter)	Data type
Type (rdoQuery)	Query function
UpdateCriteria	Batch update WHERE clause
UpdateOperation	Batch update operation
UpdateReturnCode	Batch update completion status
Validate	Data validation constants

Attributes Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstAttributesPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used with the **Attributes** property to determine the characteristics of specific **rdoColumn** objects.

Constant	Value	Description
rdFixedColumn	1	The column size is fixed (default for numeric columns). For character columns this indicates the column is defined as **rdTypeCHAR**.
rdVariableColumn	2	The column size is variable (character columns only). Indicates the column is defined as **rdTypeVARCHAR**.
rdAutoIncrColumn	16	The column value for new rows is automatically incremented to a unique **Long** integer that can't be changed.
rdUpdatableColumn	32	The column value can be changed.
rdTimestampColumn	64	The column is defined as a system-managed TimeStamp.

BOFAction Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstBOFActionPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used with the **RemoteData** control's **BOFAction** property to determine how RDO behaves when the **rdoResultset** object's **BOF** property changes state.

Constant	Value	Description
rdMoveFirst	0	**MoveFirst** (Default): Keeps the first row as the current row.
rdBOF	1	**BOF**: Moving past the beginning of an **rdoResultset** triggers the **RemoteData** control's Validate event on the first row, followed by a Reposition event on the invalid (**BOF**) row. At this point, the Move Previous button on the **RemoteData** control is disabled.

CursorDriver, rdoDefaultCursorDriver Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstCursorDriverC;vbproBooksOnlineJumpTopic"}

These constants are used to determine which type of RDO cursor library is chosen by the ODBC Driver Manager.

Constant	Value	Description
rdUseIfNeeded	0	The ODBC driver will choose the appropriate style of cursors. Server-side cursors are used if they are available.
rdUseOdbc	1	The RDO layer will use the ODBC cursor library. This gives better performance for small result sets but degrades quickly for larger result sets.
rdUseServer	2	Use server-side cursors. For most large operations this will give better performance, but can cause more network traffic.
rdUseClientBatch	3	Use the client batch cursor library used for batch update operations.
rdUseNone	4	Do not open a cursor. Create a forward-only, read-only single-row result set. This uses data access fetch techniques similar to those used by VBSQL and ODBC API access methods.

Direction Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstDirectionPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how individual parameters are to be handled when used in parameter queries.

Constant	Value	Description
rdParamInput	0	(Default) The parameter is used to pass information to the procedure.
rdParamInputOutput	1	The parameter is used to pass information both to and from the procedure.
rdParamOutput	2	The parameter is used to return information from the procedure as in an output parameter in SQL.
rdParamReturnValue	3	The parameter is used to pass the return value from a procedure.

EditMode Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstEditModeC;vbproBooksOnlineJumpTopic"}

These constants are used to determine the state of the RDO **EditMode** property.

Constant	Value	Description
rdEditNone	0	No editing operation is in progress.
rdEditInProgress	1	The **Edit** method has been invoked, and the current row is in the copy buffer.
rdEditAdd	2	The **AddNew** method has been invoked, and the current row in the copy buffer is a new row that hasn't been saved in the database.

EOFAction Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstEOFActionC;vbproBooksOnlineJumpTopic"}

These constants are used with the **RemoteData** control's **EOFAction** property to determine how RDO behaves when the **rdoResultset** object's **EOF** property changes state.

Constant	Value	Description
rdMoveLast	0	**MoveLast** (Default): Keeps the last row as the current row.
rdEOF	1	**EOF**: Moving past the end of an **rdoResultset** triggers the **RemoteData** control's Validation event on the last row, followed by a Reposition event on the invalid (**EOF**) row. At this point, the Move Next button on the **RemoteData** control is disabled.
rdAddNew	2	**AddNew**: Moving past the last row triggers the **RemoteData** control's Validation event to occur on the current row, followed by an automatic **AddNew**, followed by a Reposition event on the new row.

Error Event Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstErrorC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO should handle the message generated by the Error event. They are applied to the ***CancelDisplay*** argument.

Constant	Value	Description
rdDataErrContinue	0	Continue and do not auto-display the error message.
rdDataErrDisplay	1	(Default) Display the error message.

Prompt Property, OpenConnection, EstablishConnection Method Prompt Argument Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstOpenConnectionMethodPromptArgumentC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how the ODBC Driver Manager prompts the user when a connection is established using the **OpenConnection** or **EstablishConnection** methods, or when using the **RemoteData** Control.

Constant	Value	Description
rdDriverPrompt	0	The driver manager displays the ODBC (Open Database Connectivity) Data Sources dialog box. The connection string used to establish the connection is constructed from the data source name (DSN) selected and completed by the user via the dialog boxes. Or, if no DSN is chosen and the **DataSourceName** property is empty, the default DSN is used.
rdDriverNoPrompt	1	The driver manager uses the connection string provided in *connect*. If sufficient information is not provided, the **OpenConnection** method returns a trappable error.
rdDriverComplete	2	If the connection string provided includes the DSN keyword, the driver manager uses the string as provided in *connect*, otherwise it behaves as it does when **rdDriverPrompt** is specified.
rdDriverCompleteRequired	3	(Default) Behaves like **rdDriverComplete** except the driver disables the controls for any information not required to complete the connection.

OpenResultset Method LockType Argument, LockType Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstOpenResultMethodLockTypeC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO manages concurrency on **rdoResultset** objects created with the **OpenResultset** method or the **RemoteData** control.

Constant	Value	Description
rdConcurReadOnly	1	(Default) Cursor is read-only. No updates are allowed.
rdConcurLock	2	Pessimistic concurrency. Cursor uses the lowest level of locking sufficient to ensure the row can be updated.
rdConcurRowVer	3	Optimistic concurrency based on row ID. Cursor compares row ID in old and new rows to determine if changes have been made since the row was last accessed.
rdConcurValues	4	Optimistic concurrency based on row values. Cursor compares data values in old and new rows to determine if changes have been made since the row was last accessed.
rdConcurBatch	5	Optimistic batch mode operation. Can only be used when **CursorDriver** is set to **rdUseClientBatch**.

OpenResultset Method Type Argument, ResultsetType and Type Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstOpenResultsetMethodTypeArgumentC;vbproBooksOnlineJumpTopic"}

These constants are used to determine the type of cursor created by the selected cursor library.

Constant	Value	Description
rdOpenForwardOnly	0	Opens a forward-only – type **rdoResultset** object. (Default)
rdOpenKeyset	1	Opens a keyset-type **rdoResultset** object.
rdOpenDynamic	2	Opens a dynamic-type **rdoResultset** object.
rdOpenStatic	3	Opens a static-type **rdoResultset** object.

Options Property, Execute and OpenResultset Methods Options Argument Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstOptionsPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO should process the **Execute** and **OpenResultset** methods, and build **rdoResultset** objects using the **RemoteData** control.

Constant	Value	Description
rdAsyncEnable	32	Execute operation asynchronously.
rdExecDirect	64	Execute query using SQLExecDirect instead of SQLPrepare/ SQLExecute. The **rdoQuery** object's **Prepared** property also controls this feature.
rdFetchLongColumns	128	Download all the data for long character and long binary columns.

rdoLocaleID Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstRdoLocaleIDPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used to determine which language RDO should use when generating error messages.

Constant	Value	Description
rdLocaleSystem	0	System
rdLocaleEnglish	1	English
rdLocaleFrench	2	French
rdLocaleGerman	3	German
rdLocaleItalian	4	Italian
rdLocaleJapanese	5	Japanese
rdLocaleSpanish	6	Spanish
rdLocaleChinese	7	Chinese
rdLocaleSimplifiedChinese	8	Simplified Chinese
rdLocaleKorean	9	Korean

SqlRetCode Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstSqlRetCodePropertyC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how the latest RDO operation behaved as indicated in the **rdoError** object.

Constant	Value	Description
rdSQLSuccess	0	The operation is successful.
rdSQLSuccessWithInfo	1	The operation is successful, and additional information is available.
rdSQLNoDataFound	100	No additional data is available.
rdSQLError	-1	An error occurred performing the operation.
rdSQLInvalidHandle	-2	The handle supplied is invalid.

Status Property, rdoColumn Object Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstStatusPropertyRdoColumnObjectC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how specific **rdoColumn** objects were affected by the last **BatchUpdate** method.

Constant	Value	Description
rdColUnmodified	0	The column has not been modified or has been updated successfully.
rdColModified	1	The column has been modified.

Status Property, rdoResultset Object Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstStatusPropertyRdoResultsetObjectC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how specific **rdoResultset** rows were affected by the last **BatchUpdate** method.

Constant	Value	Description
rdRowUnmodified	0	The row has not been modified or has been updated successfully.
rdRowModified	1	The row has been modified but has not been updated in the database.
rdRowNew	2	The row has been inserted with the **AddNew** method but not yet inserted into the database.
rdRowDeleted	3	The row has been deleted but not yet deleted in the database.
rdRowDBDeleted	4	The row has been deleted locally *and* in the database.

Type Property, rdoColumn, rdoParameter Objects Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstTypePropertyRdoColumnRdoParameterObjectsC;vbproBooksOnlineJumpTopic"}

These constants are used to determine the datatype of specific columns of an **rdoResultset** object.

Constant	Value	Description
rdTypeCHAR	1	Fixed-length character string of length *n* (1 n 254). Length set by **Size** property
rdTypeNUMERIC	2	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeDECIMAL	3	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeINTEGER	4	Signed, exact numeric value with precision 10, scale 0 (signed: -231 n 231-1; unsigned: 0 n 232-1).
rdTypeSMALLINT	5	Signed, exact numeric value with precision 5, scale 0 (signed: -32,768 n 32,767, unsigned: 0 n 65,535).
rdTypeFLOAT	6	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10-308 to 10308).
rdTypeREAL	7	Signed, approximate numeric value with mantissa precision 7 (zero or absolute value 10-38 to 1038).
rdTypeDOUBLE	8	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10-308 to 10308).
rdTypeDATE	9	Date — data source dependent.
rdTypeTIME	10	Time — data source dependent.
rdTypeTIMESTAMP	11	TimeStamp — data source dependent.
rdTypeVARCHAR	12	Variable-length character string. Maximum length is DBMS-dependent.
rdTypeLONGVARCHAR	-1	Variable-length character string. Maximum length determined by data source.
rdTypeBINARY	-2	Fixed-length binary data. Maximum length is DBMS-dependent.
rdTypeVARBINARY	-3	Variable-length binary data. Maximum length is DBMS-dependent.
rdTypeLONGVARBINARY	-4	Variable-length binary data. Maximum data source dependent.
rdTypeBIGINT	-5	Signed, exact numeric value with precision 19 (signed) or 20 (unsigned), scale 0; (signed: -263 n 263-1; unsigned: 0 n 264-1).
rdTypeTINYINT	-6	Signed, exact numeric value with

		precision 3, scale 0; (signed: -128 n 127, unsigned: 0 n 255).
rdTypeBIT	-7	Single binary digit.

Type Property, rdoQuery Object Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstQueryTypeC;vbproBooksOnlineJumpTopic"}

These constants are used to determine the function of a specific **rdoQuery** object.

Constant	Value	Description
rdQSelect	0	The query contains one or more select statement
rdQAction	1	The query contains one or more Update, Insert or Delete statement
rdQProcedures	2	The query is one or more stored procedure calls
rdQCompound	3	The query contains both action and select statements

UpdateCriteria Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstUpdpateCriteriaPropertyConstantsC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO manages update operations when the **BatchUpdate** method is used.

Constant	Value	Description
rdCriteriaKey	0	Uses *just* the key column(s) in the SQL WHERE clause.
rdCriteriaAllCols	1	Uses the key column(s) *and* all updated columns in the SQL WHERE clause.
rdCriteriaUpdCols	2	Uses the key column(s) and *all* columns in the SQL WHERE clause.
rdCriteriaTimeStamp	3	Uses just the timestamp column in the SQL WHERE clause. (Generates a trappable error if no timestamp column is available).

UpdateOperation Property Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstUpdpateOperationPropertyC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO manages update operations when the **BatchUpdate** method is used.

Constant	Value	Description
rdOperationUpdate	0	Uses an Update statement for each modified row
rdOperationDelIns	1	Uses a pair of Delete and Insert statements for each modified row

ReturnCode Argument Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstUpdpateReturnCodePropertyConstantsC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how RDO manages update operations when the WillUpdateRows event is fired. The ReturnCode parameter is used to notify RDO about what your code did in the event procedure.

Constant	Value	Description
rdUpdateSuccessful	0	The developer handled the update and was successful in doing so.
rdUpdateWithCollisions	1	The developer handled the update, was successful, but some rows produced collisions (batch mode only).
rdUpdateFailed	2	The developer attempted to handle the update, but encountered an error when doing so.
rdUpdateNotHandled	3	The developer did not handle the update, RDO should continue notifying, and if no one handles the update RDO should update the data itself.

Validate Event Action Argument Constants

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdcstValidateEventActionArgumentConstantsC;vbproBooksOnlineJumpTopic"}

These constants are used to determine how the **RemoteData** control manages validation.

Constant	Value	Description
rdDataActionCancel	0	Cancel the operation when the **Sub** exits.
rdDataActionMoveFirst	1	**MoveFirst** method.
rdDataActionMovePrevious	2	**MovePrevious** method.
rdDataActionMoveNext	3	**MoveNext** method.
rdDataActionMoveLast	4	**MoveLast** method.
rdDataActionAddNew	5	**AddNew** method.
rdDataActionUpdate	6	**Update** operation (not **UpdateRow**).
rdDataActionDelete	7	**Delete** method.
rdDataActionBookmark	8	The **Bookmark** property has been set.
rdDataActionClose	9	The **Close** method.
rdDataActionUnload	10	The form is being unloaded.

action query

An SQL query that changes the underlying data or performs some administrative operation, such as adding new tables or users. An action query returns the number of rows affected rather than a result set.

aggregate function

A function, such as **Count**, **Avg**, or **Sum**, used in a query that calculates values. In writing SQL expressions, you can use SQL aggregate functions to determine various statistics.

alias

In Visual Basic, an alternate name you give to an external procedure to avoid conflict with a Visual Basic keyword, global variable, constant, or a name not allowed by the standard naming conventions.

In SQL, an alternate name you give to a column or expression in a SELECT statement, to make it shorter or more meaningful, or to prevent name conflicts when performing SQL queries using expressions that don't return names, or in a query that references the same table more than once.

base table

A table in a remote database. You can manipulate the structure of a base table using data definition SQL statements, and you can modify data in a base table using **rdoResultset** objects or action queries.

bookmark

A system-generated value identifying the current row that is contained in an **rdoResultset** object's **Bookmark** property. If you assign the **Bookmark** property value to a variable and then move to another row, you can make the earlier row current again by assigning the value of the variable to the **Bookmark** property.

Boolean

A **True**/**False** or yes/no value. Boolean values are usually stored in **Bit** columns in a remote database; however, some data sources don't support this data type directly.

Byte data type

A fundamental data type used to hold small positive integer numbers ranging from 0 to 255.

bound control

A data-aware control that can provide access to a specific column or columns in a data source through a **RemoteData** or **Data** control. A data-aware control can be bound to a **RemoteData** or **Data** control through its **DataSource** and **DataField** properties. When a **RemoteData** or **Data** control moves from one row to the next, all bound controls connected to the **RemoteData** or **Data** control change to display data from columns in the current row. When users change data in a bound control and then move to a different row, the changes are automatically saved in the data source.

Cartesian product

The result of joining two relational tables, producing all possible ordered combinations of rows from the first table with all rows from the second table.

Generally, a Cartesian product results from executing an SQL SELECT statement referencing two or more tables in the FROM clause, and not including a WHERE or JOIN clause that indicates how the tables are to be joined.

case-sensitive

Capable of distinguishing between uppercase and lowercase letters. A case-sensitive search finds text that is an exact match of uppercase and lowercase letters. Such a search would, for instance, treat "ZeroLengthStr" and "zerolengthstr" as different. Case sensitivity is a feature of some database management systems.

column

Defines the data type, size, and other attributes of one column of an **rdoTable** or **rdoResultset**. All columns taken as a set define a row in the database. An individual column contains data related in type and purpose throughout the table; that is, a column's definition doesn't change from row to row.

commit

To accept a pending transaction. If you use transaction processing and begin a transaction, none of the changes made in the transaction will be written to the data source until you commit (accept) the transaction.

copy buffer

A location created by the **rdoResultset** object for the contents of a row that is open for editing. The **Edit** method copies the current row to the copy buffer; the **AddNew** method clears the buffer for a new row; and the **Update** method saves the data from the copy buffer to the data source, replacing the current row or inserting the new row. Any statement that resets or moves the current row pointer, or cancels the edit, will discard the copy buffer.

connect string

A string used to specify a data source and other information, such as user name and password. The connect string is usually assigned to the **Connect** property of an **rdoConnection** object or **RemoteData** control, or as an argument to the **OpenConnection** method.

criteria

A set of limiting conditions, such as **=** *"Denmark"* (meaning equal to Denmark) or > *30000*, used in creating a query or filter to show a specific set of rows.

current transaction

All changes made to an **rdoResultset** object or a set of rows in a database after you use the last **BeginTrans** method and before you use the **RollbackTrans** or **CommitTrans** method.

remote data object

An object, such as **rdoConnection**, **rdoTable**, **rdoResultset**, or **rdoQuery**, that represents an object used to organize and manipulate data in code.

data page

A portion of the database in which row data is stored. Depending on the size of the rows, a data page may contain more than one row. A data page in most remote databases is 2K bytes.

data type (Remote Data)

The attribute of a variable or column that determines what kind of data it can hold. For example, an **rdoColumn** defined with the **rdTypeCHAR** data type is designed to contain text.

database

A set of data related to a particular topic or purpose. A database contains tables and can also contain queries and table relationships, as well as table and column validation criteria.

database engine

The part of the database management system that retrieves data from and stores data in user and system databases.

dissociate result set

An **rdoResultset** object which is not associated with a specific connection. Basically, a dissociate result set becomes a static snapshot of data. It is updatable, but changes are not posted to the remote database until the **rdoResultset** is re-associated with a specific connection.

rdoConnection object (definition)

Represents an open connection to a remote data source.

database management system (DBMS)

Software used to organize, analyze, and modify information stored in a database. For example, the Microsoft SQL Server is an example of a database management system.

RemoteData control

Provides access to data stored in a remote ODBC data source. The **RemoteData** control allows you to move from row to row in a result set and to display and manipulate data from the rows using bound controls.

data-definition query

An SQL-specific query that can create, alter, or delete a table, or create or delete an index in a database.

data source

A named Open Database Connectivity (ODBC) resource that specifies the location, driver type, and other parameters needed by an ODBC driver to access a database.

database administration

Activities required to created, manage and preserve the integrity and security of a database, such as maintaining user permissions and backing up and repairing the database.

default environment

The **rdoEnvironment** object that Visual Basic automatically establishes when your application first references any remote data object. This **rdoEnvironment** is referenced by **rdoEngine.rdoEnvironments**(0) or simply **rdoEnvironments**(0).

expression (Remote Data)

Any combination of operators, constants, literal values, functions, and names of columns, controls, and properties that evaluates to a single value. You can use expressions as settings for many properties and action arguments, to set criteria or define calculated columns in queries.

column properties

Attributes of a column that describe the data it contains. **Size** and **Type** are examples.

filter

A set of criteria applied to rows in order to create a subset of the rows.

forward scroll

Movement toward the end (EOF) of an **rdoResultset** object.

forward-only-type rdoResultset

An **rdoResultset** object in which rows can be searched only from beginning to end; the current row position can't be moved back toward the first row, and only one row at a time is accessible. Forward – only – type **rdoResultset** objects are useful for quickly retrieving and processing data.

identifier

An element of an expression that refers to the value of a column or property.

index (Remote Data)

A dynamic cross-reference of one or more table data columns that permits faster retrieval of specific rows from a table. As rows are added, changed, or deleted, the database management system automatically updates the index to reflect the changes.

initialization file

An ASCII text file used to contain parameters for configuring Windows-based applications or Microsoft Windows itself. Generally, an initialization file uses the extension .ini and is named after the executable program that uses it. For example, a program named Testing.exe would expect an initialization file called Testing.ini.

Integer data type

A fundamental data type that holds integer numbers. An **Integer** variable is stored as a 16-bit (2-byte) number ranging in value from -32,768 to 32,767. The type-declaration character is **%** (ANSI character 37).

locked

The condition of a data page or row that makes it read-only to all users except the one who is currently entering data in it.

ODBC (Open Database Connectivity)

A standard protocol that permits applications to connect to a variety of external database servers or files. ODBC drivers used by the ODBC driver manager permit access to SQL Server and several other data sources, including text files and Microsoft Excel spreadsheets.

optimistic

A type of locking in which the data page containing one or more rows, including the row being edited, is unavailable to other users only while the row is being updated by the **Update** method, but is available between the **Edit** and **Update** methods. Optimistic locking is used when the **rdConcurRowver** or **rdConcurValues LockType** is used when opening an **rdoResultset**.

parameter

An element containing a value that you can change to affect the results of the query. For example, a query returning data about an employee might have a parameter for the employee's name. You can then use one **rdoParameter** of the **rdoQuery** object to find data about any employee by setting the parameter to a specific name before running the query.

pessimistic

A type of locking in which each page touched by the current rowset (as determined by the **RowsetSize** property) is locked as soon as the cursor is opened and not freed until the rowset is repositioned – when the next set of pages touched by the current rowset is locked. Pessimistic locking is enabled when the **rdConcurLock** ***LockType*** option is used when opening an **rdoResultset**. This type of locking is used only in special circumstances.

query

A formalized instruction to a database to either return a set of rows or perform a specified action on a set of rows, as specified in the query. For example, the following SQL query statement returns rows:

```
SELECT CompanyName FROM Publishers WHERE State = 'NY'
```

Query types include select, action, parameter, combination, and stored procedure queries.

read-only

A type of access to data whereby information can be retrieved but not modified. This will provide better performance in most cases.

row

A set of related data about a person, place, event, or some other item. Table data is stored in rows in the database. Each row is composed of a set of related columns — each column defining one attribute of information for the row. Taken together, a row defines one specific unit of retrievable information in a database.

requery

To rerun a query to reflect changes to the rows, retrieve newly added rows, and eliminate deleted rows.

security

Used to specify or restrict the access that specified users or user groups have to data and objects in a database.

server

The database management system designed to share data with client applications; servers and clients are often connected over a network. A database server usually contains and manages a central repository of data that remote client applications can retrieve and manipulate.

SQL statement

An expression that defines a Structured Query Language (SQL) command, such as SELECT, UPDATE, or DELETE, and which might include clauses such as WHERE and ORDER BY. SQL strings and statements are typically used in queries and **rdoResultset** objects but can also be used to create or modify a database structure.

The syntax for SQL statements is dependent on the data source.

table

A basic unit of data storage in a relational database. A table stores data in rows and columns and usually contains a particular category of things, such as employees or parts. Also called a base table.

rdoTable object

Represents the stored definition of a base table or an SQL view.

transaction

A series of changes made to a database's data. Mark the beginning of a transaction with the **BeginTrans** statement, commit the transaction using the **CommitTrans** statement, or undo all your changes since **BeginTrans** using the **RollbackTrans** statement.

Transactions are optional, but can increase the speed of operations and allow changes to be reversed.

Transactions can be managed at the **rdoConnection** level or at the **rdoEnvironment** level.

Transactions can also be managed by a Distributed Transaction Coordinator.

update

The process that saves changes to data in a row. Until the row is saved, changes are stored in a temporary row called the copy buffer.

The UPDATE clause in an SQL statement changes data values in one or more rows in a database table.

message

A packet of information passed from one application to another.

multiuser database

A database that permits more than one user to access and modify the same set of data at the same time. In some cases, the additional "user" may be another instance of your application, or another application running on your system that accesses the same data as some other application.

normalize

To minimize the duplication of information in a relational database through effective table design.

null

A value that indicates missing or unknown data. **Null** values can be entered in columns for which information is unknown and in expressions and queries. In Visual Basic, the **Null** keyword indicates a **Null** value.

null column

A column containing no characters or values. A null column isn't the same as a zero-length string ("") or a column with a value of 0. A column is set to null when the content of the column is unknown. For example, a Date_Completed column in a task table would be left null until a task is completed.

ODBC driver

A dynamic-link library (DLL) used to connect a specific Open Database Connectivity data source with a client application. For example, there are specific drivers for Microsoft SQL Server included with Visual Basic. To work with RDO, ODBC drivers must comply with ODBC Level II requirements.

optimistic batch

An optimistic batch is a set of rows submitted to the remote server for processing as a unit of work. In this case it is assumed that there is little possibility of update or insert collisions.

parameter query

A query that requires you to provide one or more criteria values, such as Redmond for City, before the query is run. A parameter query isn't, strictly speaking, a separate kind of query; rather, it extends the flexibility of other queries.

parse

To identify the parts of a statement or expression and then validate those parts against the appropriate language rules.

permission

One or more attributes that specify what kind of access a user has to data or objects in a database. For example, a table or query with Read Only permission permits a user to retrieve but not edit data in the table or query.

rowset population

The process of loading **rdoResultset** rows into memory.

The **rdoResultset** objects populate the number of rows defined by the **RowsetSize** attribute. If you are using server-side cursors, only this number of rows is present in memory at any given time.

session

A session begins when a user connects to a data source and ends when a user disconnects. All operations performed during a session are subject to permissions determined by the login user name and password. Sessions are implemented as **rdoConnection** objects and are synonymous with connections.

Single data type

A fundamental data type that holds single-precision floating-point numbers in IEEE format. A **Single** variable is stored as a 32-bit (4-byte) number ranging in value from -3.402823E38 to -1.401298E-45 for negative values, from 1.401298E-45 to 3.402823E38 for positive values, and 0. The type-declaration character is **!**.

Structured Query Language (SQL)

A language used in querying, updating, and managing relational databases. SQL can be used to retrieve, sort, and filter specific data to be extracted from the database.

SQL database

A database that can be accessed through the use of Open Database Connectivity (ODBC) data sources or another interface native to the database. Also known as a relational database.

SQL-specific query

A query that can be created only by writing an SQL statement.

stand-alone object

In RDO, you can create **rdoConnection** and **rdoQuery** objects using:

```
Dim X as New rdoxxxx
```

When this is done, a stand-alone object is created. Those object properties that do not depend on access to a connection or other objects can be manipulated.

temporary disk

The directory identified by the TEMP operating system environment variable. Also known as temporary drive. Although the TEMP environment variable may point to a RAM disk, this isn't recommended.

TEMP

A TEMP environment variable is initialized by your system when it is started. Generally, TEMP points to an area on your hard disk used by Microsoft Windows and other programs to store information that doesn't need to be saved after you shut down your system. For example, the following line in your autoexec.bat file points the TEMP environment variable to the D:\Temparea directory:

```
SET TEMP=D:\TEMPAREA
```

update query

An action query that changes base table data according to criteria you specify. An update query doesn't return any rows, but it does return the number of rows affected.

user account

An account identified by a user name and password that is created to manage access to objects in a remote database.

validation

The process of checking whether entered data meets certain conditions or limitations.

WHERE clause

The part of an SQL statement that specifies which rows to retrieve. The WHERE clause limits the scope of the query and specifies which columns are used to join multiple tables.

Yes/No data type

A column data type that contains a **Boolean** (**True**/**False** or yes/no) value.

zero-length string

A string containing no characters. The **Len** function of a zero-length string returns 0.

DDL (Data Definition Language)

The language used to describe, change, or define attributes of a database, especially the schema associated with tables, columns, and storage strategy.

ODBC data source

A database or database server used as a source of data. ODBC data sources are referred to by their Data Source Name or by specific reference to the ODBC driver and server name. Named Data sources can be registered using either the ODBC Administrator in the Windows Control Panel or the **rdoRegisterDataSource** method.

reserved word

A word that is part of the data source SQL language. Reserved words include the names of statements, predefined functions and data types, methods, operators, and objects. Examples include SELECT, UPDATE, BETWEEN, SET, and INSERT.

string expression

Any expression that evaluates to a sequence of contiguous characters. Elements of the expression can include a function that returns a string, a string literal, a string constant, a string variable, a string **Variant**, or a function that returns a string **Variant** (**VarType** 8).

Long data type

A four-byte integer (a whole number between -2,147,483,648 and 2,147,483,647, inclusive).

numeric expression

Any expression that can be evaluated as a number. Elements of the expression can include any combination of keywords, variables, constants, and operators that result in a number.

Object Browser

A dialog box that lets you examine the contents of an object library to get information about the objects provided, their methods and properties, and possibly their constants.

object library

A file with the .olb extension that provides information to Automation clients (like Visual Basic) about available ActiveX objects. You can use the Object Browser to examine the contents of an object library to get information about the objects provided.

ASCII character set

American Standard Code for Information Interchange (ASCII) 7-bit character set widely used to represent letters and symbols found on a standard U.S. keyboard. The ASCII character set is the same as the first 128 characters (0 – 127) in the ANSI character set.

column data types

The following table describes the column data types.

Column data type	Description
rdTypeCHAR	Fixed-length character string. Length set by **Size** property.
rdTypeNUMERIC	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeDECIMAL	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeINTEGER	Signed, exact numeric value with precision 10, scale 0 (signed: -2^31 n 2^31-1; unsigned: 0 n 2^32-1).
rdTypeSMALLINT	Signed, exact numeric value with precision 5, scale 0 (signed: -32,768 n 32,767, unsigned: 0 n 65,535).
rdTypeFLOAT	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10^-308 to 10^308).
rdTypeREAL	Signed, approximate numeric value with mantissa precision 7 (zero or absolute value 10^-38 to 10^38).
rdTypeDOUBLE	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10^-308 to 10^308).
rdTypeDATE	Date — data source dependent.
rdTypeTIME	Time — data source dependent.
rdTypeTIMESTAMP	TimeStamp — data source dependent.
rdTypeVARCHAR	Variable-length character string. Maximum length is the value returned by the Size property.
rdTypeLONGVARCHAR	Variable-length character string. Maximum length determined by data source.
rdTypeBINARY	Fixed-length binary data. Maximum length is the value returned by the Size property.
rdTypeVARBINARY	Variable-length binary data. Maximum length 255.
rdTypeLONGVARBINARY	Variable-length binary data. Maximum data source dependent.
rdTypeBIGINT	Signed, exact numeric value with precision 19 (signed) or 20 (unsigned), scale 0 (signed: -2^63 n 2^63-1; unsigned: 0 n 2^64-1).
rdTypeTINYINT	Signed, exact numeric value with precision 3, scale 0 (signed: -128 n 127, unsigned: 0 n 255).
rdTypeBIT	Single binary digit.

current row

The row in an **rdoResultset** object that you can use to modify or examine data. Use the *Move* methods to reposition the current row in a rowset.

Only one row in an **rdoResultset** can be the current row; however, an **rdoResultset** may have no current row. For example, after the current **rdoResultset** row has been deleted, or when an **rdoResultset** has no rows, the current row is undefined. In this case, operations that refer to the current row result in a trappable error.

data manipulation language (DML)

The SQL statement properties and methods you use to write applications or queries that access and manipulate the data in existing databases. This includes facilities for querying the database, navigating through its tables, performing updates, and adding or deleting rows.

Date/Time

Dates and times are stored internally as different parts of a real number.

The value to the left of the decimal represents a date between January 1, 100 and December 31, 9999, inclusive. Negative values represent dates prior to December 30, 1899.

The value to the right of the decimal represents a time between 0:00:00 and 23:59:59, inclusive. Midday is represented by .5.

design time

The time during which you build an application in the development environment by adding controls, setting control or form properties, and so on. In contrast, during run time, you interact with the application as a user would.

environment

An **rdoEnvironment** object defines a session for a specific user. When RDO is referenced for the first time, a default **rdoEnvironment** object is created with a password of "" and user name of "".

object expression

An expression that specifies a particular object. This expression can include any of the object's containers.

rdoPreparedStatement object

A remote data object that contains a prepared SQL statement and collection of **rdoParameter** objects for each parameter in the **rdoPreparedStatement**. This object is obsolete and should be replaced with the **rdoQuery** object.

rdoQuery object (definition)

A remote data object that contains an SQL statement and collection of **rdoParameter** objects for each parameter in the **rdoQuery**. An **rdoQuery** object is used to manage parameterized queries or queries used repeatedly throughout execution of an application.

Parameterized Queries

A query that requires one or more parameters or arguments before execution. For example, an SQL statement that refers to a specific part number as supplied by the user would use a parameterized query to insert the user-provided number into the SQL statement before execution so that the query references that specific part.

run time

The time when an application is running. During run time, you interact with the code as a user would. In contrast, design time is when the application is developed.

server-side cursor

Cursor keysets that are created on the server instead of on the client workstation.

sort order

A sequencing principle used to order data, such as dictionary, binary, ascending, descending, and so on.

SQL view

SQL views are similar to queries: both allow you to limit the rows and columns displayed from one or more tables, and both provide similar functionality. SQL views are logical sets of rows where a table represents the actual rows.

statement handle

Used by the ODBC driver to reference storage for names, parameter and binding information, error messages, and other information related to a statement processing stream. The **hStmt** property of the **rdoResultset** is an ODBC statement handle.

stored procedure

A pre-compiled procedure stored in a data source, available to be called from an application as needed. Predefined queries reduce the overhead of repeatedly specifying the same selection criteria, and are much faster than submitting an ad-hoc query.

two-phase commit

Allows an application to coordinate updates among multiple SQL servers. This implementation of distributed transactions treats transactions on separate SQL servers as a single transaction. The service uses one SQL server, the commit server, as a record keeper that helps the application determine whether to commit or to roll back transactions. Thus, the two-phase commit guarantees that either all the databases on the participating servers are updated or that none of them are.

Variant data type

A special data type that can contain numeric, string, or date data, as well as the special values **Empty** and **Null**. The **VarType** function defines how the data in a **Variant** is treated. All variables become variant types if not explicitly declared as some other type.

asynchronous

A type of query mode in which SQL queries return immediately, even though the results are still pending. This enables an application to continue with other processing while the query is pending completion.

cursor

A logical set of rows managed by the data source or ODBC driver manager. The cursor is so named because it indicates the current position in the result set, just as the cursor on a CRT screen indicates current position.

connection handle

Identifies memory storage for information about a particular connection. RDO will request a connection handle prior to connecting to a data source; RDO manages connection handles automatically through the **rdoConnection** object. Each connection handle is associated with an environment handle. An environment handle can have multiple connection handles associated with it, and there can be multiple environments.

Data Access Objects (DAO)

Objects that are defined by the Microsoft Jet database engine. You use data access objects, such as the **Database**, **TableDef**, **Recordset**, and **QueryDef** objects, to represent objects that are used to organize and manipulate data in code.

dynamic-link library (DLL)

A library of routines loaded and linked to applications at run time.

dynamic-type rdoResultset

The result of a query that can have updatable rows. A dynamic-type **rdoResultset** is a dynamic set of rows that you can use to add, change, or delete rows from an underlying database table or tables. A dynamic-type **rdoResultset** can contain columns from one or more tables in a database. Membership of a dynamic **rdoResultset** is not fixed.

environment handle

Identifies memory storage for global information, including the valid connection handles and current active connection handle. RDO will request this handle prior to connecting to a data source. The remote data objects manage environment handles automatically through the **rdoEnvironment** object.

escape codes

Allow you to specify a value such as a date or time, in a data-independent way. For example, `{d 'value'}` allows you to specify the date `(SELECT * FROM table WHERE DateField = {d "2/17/94"})`. When this query is submitted to the ODBC driver, it will scan the string and replace the escape clause with the date in the proper form for the specific ODBC driver you are using.

keyset

The set of key values used to identify specific rows in a cursor. Keysets are stored on the server in a server-side keyset cursor and on the workstation on client-side keyset cursors.

keyset-type rdoResultset

The result of a query that can have updatable rows. Movement within the keyset is unrestricted. A keyset-type **rdoResultset** is a set of rows that you can use to add, change, or delete rows from an underlying database table or tables. A keyset-type **rdoResultset** can contain columns from one or more tables in a database. Membership in a keyset **rdoResultset** is fixed.

native error

An error generated and returned from the database management system of the data source on a given connection.

ODBC driver manager

Provides the interface from the host language to the specific back-end data source driver.

Remote Data Objects (RDO)

Provide an information model for accessing remote data sources through ODBC. RDO offers a set of objects that make it easy to connect to a database, execute queries and stored procedures, manipulate results, and commit changes to the server.

scope

Defines the visibility of a variable, procedure, or object. For example, a variable declared as **Public** is visible to all procedures in all modules. Variables declared in procedures are visible only within the procedure and lose their value between calls unless they are declared **Static**.

static-type rdoResultset

The membership, order, and values in a result set used by a static cursor are generally fixed when the cursor is opened. Rows updated, deleted, or inserted by other users are not detected by the cursor until it is closed and then reopened or required.

timestamp

Contains a unique value that is updated automatically whenever a row is updated.

procedural query

An SQL query that executes a stored procedure.

connection

A link to an ODBC data source.

compile time

The moment at which source code is translated into executable code.

select query

A query that asks a question about the data stored in your tables, and returns an **rdoResultset** object without changing the data. Once the **rdoResultset** data is retrieved, you can examine and make changes to the data in the underlying queries.

multiple resultset query

A query that contains more than one select query and returns more than a single set of results. Multiple resultset queries can also contain a combination of select and action queries.

String data type

A fundamental data type that holds character information. A **String** variable can contain approximately 65,535 bytes (64K), is either fixed-length or variable-length, and contains one character per byte. Fixed-length strings are declared to be a specific length. Variable-length strings can be any length up to 64K, less a small amount of storage overhead.

The type-declaration character for the **String** data type is **$**.

Already beyond the end of the result set (Error 40024)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgMoveNextC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgMoveNextS"}

You attempted to call **rdoResultset.MoveNext** when the **EOF** property was set to **True**.

To avoid this error, check the state of the **EOF** property before calling **MoveNext**.

An error occurred configuring the DSN. Please check the parameters and try again (Error 40000)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgConfigDSNC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgConfigDSNS"}

The call to **rdoEngine.rdoRegisterDataSource** failed.

To avoid this error, check the validity of the **rdoRegisterDataSource** arguments passed and try again.

An error occurred loading the ODBC installation library (ODBCCP32.DLL) (Error 40032)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgLoadODBCLibC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgLoadODBCLibS"}

You attempted to call **rdoEngine.rdoRegisterDataSource** and the application could not load the ODBC installation library file ODBCCP32.DLL.

To avoid this error, make sure ODBC is correctly installed on the machine that generated the error, and that the file ODBCCP32.DLL is in the system path.

An internal ODBC error was encountered (Error 40002)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgODBCC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgODBCS"}

An ODBC error occurred on the most recently invoked property or method.

The exact error depends on the ODBC driver and type of database you are using. Examine the **rdoErrors** Collection for an exact description of the problem.

Note ODBC can generate more than one error during statement execution. Make sure you check each error in the **rdoErrors** collection.

An invalid ODBC handle was encountered (Error 40004)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidODBCC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidODBCS"}

An error caused by an invalid ODBC statement handle occurred when executing an ODBC operation.

Examine the **rdoErrors** collection for an exact description of the problem. If no information is found, make sure the statement handle wasn't deallocated or altered by a previous operation.

An invalid value for the concurrency option was passed (Error 40019)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidConcurrencyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidConcurrencyS"}

An invalid lock type was passed to either **rdoPreparedStatement.LockType** or the *locktype* argument in **rdoPreparedStatement.OpenResultset**.or the **rdoQuery**.**LockType** or the *locktype* argument in **rdoQuery.OpenResultset**.

To avoid this error, make sure you pass one of the following valid lock types:

- **rdConcurReadOnly**
- **rdConcurLock**
- **rdConcurRowVer**
- **rdConcurValues**

Note Not all lock types can be used on every data source.

An invalid value for the cursor driver was passed (Error 40003)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidCursorDriverC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidCursorDriverS"}

An invalid type value was passed to either **rdoEnvironment.CursorDriver** or **rdoEngine.rdoDefaultCursorDriver**.

To avoid this error, pass one of the following values:

- **rdUseIfNeeded**
- **rdUseODBC**
- **rdUseServer**
- **rdUseClient**
- **rdUseNone**

Note Not all data source drivers support all cursors.

An invalid value for the prompt option was passed (Error 40033)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgPromptC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgPromptS"}

You attempted to call **rdoEnvironment.OpenConnection**, and the value for the *prompt* argument was not one of the following values:

- **rdDriverPrompt**
- **rdDriverNoPrompt**
- **rdDriverComplete**
- **rdDriverCompleteRequired**

To avoid this error, make sure the *prompt* argument is one of the previously mentioned values.

An invalid value for the cursor type parameter was passed (Error 40034)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgCursorTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCursorTypeS"}

You attempted to call the **OpenResultset** method, and the value for the *type* argument was not one of the following values:

- **rdOpenKeyset**
- **rdOpenDynamic**
- **rdOpenStatic**
- **rdOpenForwardOnly**

To avoid this error, make sure the *type* argument is one of the previously mentioned values.

Note Not all data sources support all cursors.

BOF already set (Error 40025)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgBOFSetC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgBOFSetS"}

You attempted to call **rdoResultset.MovePrevious** when the **BOF** property was set to **True**. This means that the current row pointer is already positioned before the first row in the result set, and you are trying to perform a move operation that would move the row pointer to an invalid position.

To avoid this error, check the state of the **BOF** property before calling the **MovePrevious** method.

Can't create prepared statement for invalid database connection (Error 40015)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgConnectionStateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgConnectionStateS"}

The program tried to create an **rdoPreparedStatement** or **rdoQuery** object on an invalid **rdoConnection**.

To avoid this problem, make sure the **rdoConnection** object is currently connected to a data source.

Can't execute empty rdoPreparedStatement (Error 40018)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgEmptyQDC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgEmptyQDS"}

A Null or empty string was encountered in the **SQL** property of **rdoPreparedStatement** or **rdoQuery** object during the invocation of the **Execute** method.

To avoid this error, make sure the SQL statement for the **rdoPreparedStatement** or **rdoQuery** object is valid, either by passing it as an argument to **CreatePreparedStatement** or **CreateQuery**, or by setting the **SQL** property of the **rdoPreparedStatement** or **rdoQuery** object.

Can't execute unprepared rdoPreparedStatement (Error 40017)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgUnpreparedQDC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUnpreparedQDS"}

An ODBC error occurred trying to prepare the SQL statement passed in **rdoPreparedStatement.Execute** or **rdoQuery**.**Execute**

Check the **rdoErrors** collection for more detail and make sure the SQL statement for the **rdoPreparedStatement** or **rdoQuery** object is valid for the data source you are referencing.

Can't move relative to current row as EOF/BOF already set (Error 40029)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgMoveEOFBOFC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgMoveEOFBOFS"}

You attempted to call **rdoResultset.Move** specifying a relative move (by passing Null as the *bookmark* argument) when the result set is currently positioned at **EOF** or **BOF**, or the result set is marked as invalid.

Note If **EOF** or **BOF** are set, a valid bookmark must be passed as part of the call to **Move**.

To avoid this error, make sure the **EOF** or **BOF** properties are not set, or a valid bookmark is provided.

Invalid bookmark (Error 40027)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidBookmarkC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidBookmarkS"}

An invalid bookmark value was passed to **rdoResultset.Move**.

To avoid this error, be sure to pass a valid bookmark. Retrieve the bookmark by using **rdoResultset.Bookmark**, and make sure the variable you use to store the bookmark is still valid.

Invalid bookmark argument to move (Error 40028)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidBookmarkMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidBookmarkMoveS"}

The *bookmark* argument to **rdoResultset.Move** was not passed as the correct data type.

To avoid this error, be sure to pass the *bookmark* argument as either an **Integer** or **Byte** data type. Retrieve the bookmark by using **rdoResultset.Bookmark**, and be sure that the variable you use to store the bookmark is still valid.

Invalid connection string (Error 40005)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidConnectionS"}

An invalid connection string was passed to **rdoEnvironment.OpenConnection** or the **EstablishConnection** method through the Connect property of a private **rdoConnection** object.

To avoid this error, be sure to pass a valid ODBC connection string to the **OpenConnection** method of **rdoEnvironment**.

Invalid resultset state for update (Error 40026)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidUpdateStateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidUpdateStateS"}

You attempted to call **rdoResultset.Update** when the current row pointer was not pointing to a valid row, or the result set was marked as invalid. This occurs if:

- The current row pointer is pointing at **EOF** or **BOF**.
- The result set has been marked invalid due to a call to a method such as **Cancel**.
- An SQL error occurs.

Also, deleting a row will mark it as invalid.

To avoid this error, check the state of the **BOF** and **EOF** properties before calling **Update**, and make sure no method was called prior to calling **Update** that would mark the result set as invalid.

Invalid state for Move (Error 40023)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidStateMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidStateMoveS"}

An attempt was made to call either **rdoResultset.MoveFirst** or **rdoResultset.MoveNext** when the result set was marked as invalid. A result set can be marked as invalid if:

- You called the **Cancel** method on the result set prior to calling a **Move** method.
- You called the **MoreResults** method and there are no more result sets.
- An SQL error occurs.

To avoid this error, be sure that the current result set is valid and that you have not called an operation that would mark it as invalid.

Object collection: illegal modification -- collection is read-only (Error 40049)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgCollReadonlyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCollReadonlyS"}

You attempted to programmatically modify the contents of an RDO collection. This RDO collection is read-only.

To avoid this error, do not attempt to modify the contents of this type of RDO collection. For this type of collection, Items are added to the collection automatically, and they are removed when the **Close** or R**e**move methods for an object are invoked.

SQL returned No Data Found (Error 40001)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNoDataS"}

An SQL statement you tried to execute returned a message indicating no data was found for a query, or no rows were affected by the action (**Insert**, **Update**, **Delete**) statement you attempted to execute. This may be a valid response, however, in cases where you issue a query expecting no data or rows to exist for that query.

To avoid this error, change the criteria in your SQL statement and try again.

An invalid parameter was passed (Error 40054)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidParamS"}

The value passed to a property or method is not a valid value.

Most properties and methods accept values of only a certain type, within a certain range, and an inappropriate value has been assigned to a property or method. See the property or method's Help topic to determine the appropriate types and range of values.

Can't assign value to column unless in edit mode (Error 40039)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgFieldNotEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgFieldNotEditS"}

You attempted to assign a value to a column before calling **rdoResultset.Edit** or **rdoResultset.AddNew**.

To avoid this error, make sure the **Edit** or **AddNew** method has been called before assigning values to columns with either the **Value** property or the **AppendChunk** method.

Can't assign value to non-updatable column (Error 40038)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgFieldNotUpdatableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgFieldNotUpdatableS"}

You attempted to assign a value to a column that is not updatable.

To avoid this error, make sure the result set is updatable before assigning a value to a column with either the **Value** property or the **AppendChunk** method. Note that the **Updatable** property does not always reflect the updatability of any given resultset due to a number of factors including permissions and type of cursor.

Can't assign value to output-only parameter (Error 40043)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgOutputOnlyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgOutputOnlyS"}

An attempt was made to set a value for a parameter that is defined as an output (only) parameter.

To avoid this error, use the **Direction** property to be sure the parameter for which you are setting a value is defined as either an input (**rdParamInput**) or input/output (**rdParamInputOutput**) parameter. Generally, the Direction property is set automatically based on how the called procedure is defined, but some drivers require you to set the direction before accessing the parameter. This error could also occur if you mismatch the ordinal number or name of a parameter and mistake an input parameter for an output parameter.

Can't assign value to unbound column (Error 40037)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgAssignUnboundC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgAssignUnboundS"}

You attempted to assign a value to a column that represents a large binary object (BLOB) or similar object.

To avoid this error, use the **AppendChunk** method to assign values to columns of this type. Use the **ChunkRequired** property to determine if the column in question requires the use of **AppendChunk**.

Can't assign value to unbound parameter (Error 40042)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUnboundParameterS"}

An attempt was made to set a value for a parameter that has not been bound.

To avoid this error, make sure no error was returned from prior RDO methods. If an error was returned, fix the SQL statement that was passed, and try the operation again.

Column not bound correctly (Error 40035)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgFieldNotBoundC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgFieldNotBoundS"}

An attempt was made to open a result set with a column of unknown data type. In most cases, the ODBC driver is capable of recognizing all data types supported by the version of back-end that is being accessed. This error can occur when the driver does not recognize one of the returned data types.

For more information, consult the documentation for the data source from which the column data originated. For example, consult the DRVSSRVR.HLP file for detailed information on the SQL Server driver.

GetNewEnum: Couldn't get interface for IID_IUnknown (Error 40050)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNewEnumIIDS"}

An internal error occurred while attempting to allocate memory to enable the **For...Each** syntax to iterate over a collection.

This error can be caused by any condition that exhausts system resources including internal handles, RAM, disk space or object reference pointers. Watch for recursive procedures or procedures that leak memory.

Incorrect type for parameter (Error 40040)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidTypeS"}

A **Variant** with an invalid type was detected. This can be caused by passing a value to an RDO method or property that cannot be coerced to the correct type, such as trying to pass a string data type as a numeric value.

To avoid this problem, make sure the value passed is the correct type for the operation. For column values, you can check the column's **Type** property to ensure you are passing the correct type.

Object Collection: Couldn't find item indicated by text (Error 40041)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgCollecNotFoundC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCollecNotFoundS"}

You attempted to address an object in a collection by using a text value, and no object in that collection matched the text string supplied. For example, you tried to reference a column of a result set by name and the specified column does not exist in the result set.

To avoid this error, make sure an object in the collection has a **Name** property that matches the text string supplied, or use the object's ordinal value.

Object Collection: This collection doesn't support location by text tag (Error 40021)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCollecTagS"}

You attempted to find an object in a collection with a text string, and the objects in the collection do not support lookup by text strings. For example, you cannot reference members of the **rdoErrors** collection by name -- only by ordinal number.

To avoid this error, use an ordinal value instead, such as **rdoErrors**(1).

The object has already been closed (Error 40046)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgAlreadyClosedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgAlreadyClosedS"}

You have attempted to call the **Close** method on an object that has already been closed.

To avoid this problem, do not use the **Close** method on an object that has already been closed.

This environment name already exists in the collection (Error 40048)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgDupnameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgDupnameS"}

You attempted to call the **rdoEngine.rdoCreateEnvironment** method, the **rdoConnection.CreatePreparedStatement, rdoConnection.CreateQuery**, or **Add** method passing a name that already exists in the **rdoEnvironments, rdoConnections** or **rdoPreparedStatements** collection.

To avoid this error, make sure the name you pass does not conflict with a name already added to the collection.

Unbound column - use GetChunk (Error 40036)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgUnboundFieldC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUnboundFieldS"}

You attempted to access a column containing a large binary object (BLOB) or similar object.

To avoid this error, use the **GetChunk** method to access columns of this type. If this error occurs when using parameters, you cannot use parameters on columns that represent large binary objects (Text or Image columns in SQL Server). You can use the **ChunkRequired** property to determine if the column in question requires the use of **GetChunk**.

You cannot execute a query when an asynchronous query is in progress (Error 40045)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgAsyncInProgressC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgAsyncInProgressS"}

You have attempted to call a method or property while an asynchronous query is still executing.

To avoid this error, check the **StillExecuting** property, and do not call RDO methods or properties that affect the SQL statement until the **StillExecuting** property returns **False**.

You must specify a valid name for the environment (Error 40047)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidNameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidNameS"}

You have attempted to pass an invalid name for the *name* argument when calling the **rdoEngine.rdoCreateEnvironment** method or **Add** method with a private **rdoEnvironment**.

To avoid this error, make sure you pass a valid name as the *name* argument. The name can be any string expression that is not null or empty, and it should not be a duplicate of any name previously added to the collection.

An attempt was made to issue a select statement using the Execute method (Error 40057)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgExecuteSelectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgExecuteSelectS"}

You attempted to issue a SELECT statement using the **Execute** method. This error can also occur when you execute a stored procedure that contains a SELECT statement.

To issue a select query, you should instead use the **OpenResultset** method. The **Execute** method is designed for use with action queries (**Insert**, **Update**, **Delete**).

An error occurred loading the version library (VERSION.DLL) (Error 40016)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgLibraryErrorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgLibraryErrorS"}

You attempted to call **rdoEngine.rdoVersion** and the application could not load the Win32 library file VERSION.DLL.

To avoid this error, make sure the file VERSION.DLL is in the system path.

An unexpected error occurred (Error 40006)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUnexpectedErrorS"}

An unexpected error occurred that caused RDO to become unstable. This error can also occur if you use the ODBC API to manipulate RDO-generated objects using one of the ODBC handles.

To avoid this error, make sure you have enough free resources and memory, then restart the program and try again.

Incompatible data types for compare (Error 40014)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgIncompatibleCompareS"}

An **rdoResultset** was called with an argument value whose data type is not compatible with the compared column's data type.

To avoid this problem, make sure the value you are using in the comparison matches the data type of the column you are comparing against. Also, this method is valid only when called by a data source control.

Invalid operation (Error 40055)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidOperationC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidOperationS"}

The property or method invoked is not valid in this context. For example, you might have used the **Edit** or **Update** method against an **rdoResultset** that is not updatable.

To avoid this error, check the sequence of the operations you are attempting and make sure they are correct. One possible cause is that you are trying to set a column value on a column that is a meta data column (that is, a column generated from an **rdoTable** and not an **rdoResultset**).

Invalid operation for forward-only cursor (Error 40008)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidCursorOperationC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidCursorOperationS"}

The program called **rdoResultset.MovePrevious, rdoResultset.MoveLast** or **rdoResultset.MoveFirst** while processing a forward-only query.

To avoid this error, either change the cursor type to **rdOpenKeyset**, **rdOpenDynamic**, or **rdOpenStatic**, or call only **MoveNext** for a forward-only **rdoResultset**.

Invalid row for AddNew (Error 40010)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidAddNewRowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidAddNewRowS"}

There was a call to **rdoResultset.AddNew** when the cursor was positioned on an invalid row, or the program had previously called **AddNew** or **Edit** without calling **Update** or cancelling the operation.

To avoid this error, move the cursor to a valid row and make sure you have a valid result set, and the sequence of calls to **rdoResultset** are correct.

Invalid seek flag (Error 40012)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidSeekFlagS"}

An invalid seek flag was passed to **rdoResultset**. This message is only applicable to developers creating third-party bound controls.

To avoid this error, make sure the flag passed is one of the following values:

- **DBSEEK_LT**
- **DBSEEK_LE**
- **DBSEEK_EQ**
- **DBSEEK_GE**
- **DBSEEK_GT**
- **DBSEEK_PARTIALEQ**

Also, this method is only valid when called by a data source control.

No current row (Error 40009)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgNoCurrentRowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNoCurrentRowS"}

There was a call to **rdoResultset.Edit** when the cursor was positioned on an invalid row. This error may be caused by attempting to edit a deleted row, or by invoking **Edit** when the cursor is positioned either before the first row, or after the last row.

To avoid this error, move the cursor to a valid row and make sure you have a valid result set. You cannot use the Edit method if either the **BOF** or **EOF** properties are true.

Object is invalid or not set (Error 40011)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidObjectS"}

The program called a method or operation on an object that has been closed, discarded, or not allocated.

To avoid this error, make sure:

- The object has been allocated using the **Set** syntax.
- The object or its parent objects have not been closed.
- The object has not been set to **Nothing**.

Partial equality requires string column (Error 40013)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgPartialEqualityS"}

rdoResultset.FindByValues was called specifying **DBSEEK_PARTIALEQ** on a non-string column. **DBSEEK_PARTIALEQ** works only on columns that contain string data.

To avoid this error, use only **DBSEEK_PARTIALEQ** on string-based columns. Also, this method is valid only when called by a data source control.

This message is applicable to developers creating third-party bound controls.

The row you attempted to move to has been deleted (Error 40056)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRowDeletedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRowDeletedS"}

The row you attempted to move to using a bookmark has been deleted from the database. The database row in question could have been deleted by your program or some other program with access to the row.

To avoid this error, try the operation again, specifying a valid bookmark.

The rdoResultset is empty (Error 40022)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgrdoResultsetEmptyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgrdoResultsetEmptyS"}

You attempted to make a call to **rdoResultset.Move**, **rdoResultset.MoveNext**, or **rdoResultset.MovePrevious** on an empty result set.

To avoid this error, make sure the SQL statement used returns a valid result set before using any of the previously mentioned methods. You can check to see if a result set is empty by checking the **RowCount** property, or by checking to see if both the **EOF** and **BOF** properties are **True**.

The resultset is read only (Error 40058)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgTheResultsetIsReadOnly.S"}

You attempted an **Edit**, **Delete**, or **AddNew** operation on a read-only result set. Make sure you specify the correct **LockType** value that supports action queries when you open the result set.

The user canceled the operation. (Error 40059)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgErr40059S"}

The user clicked the Cancel button on an ODBC dialog box.

A control canceled the operation or an unexpected internal error has occurred.

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCCancelOperationS"}

The **RemoteData control** tried to update a row based on bound control data**,** but the **Update** operation failed. This usually occurs because of one of the following reasons:

- The data in the bound control fails validation.
- The data is not the correct data type for the result set column.
- The value in the bound control does not match the row description, rule, or trigger criteria.

Examine and modify the data values and retry the operation.

An error has occurred. Unable to retrieve error information (Error 40502)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCRetrieveErrInfoS"}

An error has occurred in the **RemoteData control**, and no detailed error information is available. This error occurs only under very unusual circumstances—never during normal operation. You'll get this error message only when the **RemoteData** control can't access detailed error information. This situation only occurs when OLE is not working properly, and is generally an indication of a more serious error condition.

An unexpected error occurred (Error 40501)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUnexpectedS"}

The **RemoteData control** tried to call an RDO method that should normally exist. The method was not found, or the call did not complete correctly. This error occurs only under very unusual circumstances—never during normal operation. You'll see this error message only when the **RemoteData** control can't get a dispatch interface to RDO, or when a method in RDO fails for an unknown reason. This error message is generally an indication of a more serious error condition.

An unexpected internal error has occurred (Error 40500)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCUnexpectedInternalErrorS"}

The **RemoteData control** attempted to notify the bound controls that new data is available, but received a failure code from RDO instead. This error occurs only under very unusual circumstances—never during normal operation—and only if something is seriously wrong with Visual Basic's binding manager or the bound controls themselves. This error message is generally an indication of a more serious error condition.

Could not refresh controls (Error 40504)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCRefreshControlC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCRefreshControlS"}

The **Refresh** method of the **RemoteData control** failed because of one of the following reasons:

- A connection could not be established.
- A result set could not be opened.
- A bound control failed to update.

Check the connection, the result set, and the control bindings (review information in Help about the bound control). This error can also occur if the server or network unexpectedly drops the connection. Generally, it indicates that the **rdoResultset** or connection is no longer usable.

Invalid object (Error 40506)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCInvalidObjectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCInvalidObjectS"}

An object other than an **rdoResultset** was assigned to the **Resultset** property. Assign a valid **rdoResultset** object to the **Resultset** property. The only object that can be assigned to the **Resultset** property of the **RemoteData** control is an **rdoResultset** object created with another **RemoteData** control or the **OpenResultset** method.

Invalid property value (Error 40505)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCInvalidPropertyS"}

An inappropriate value has been assigned to a property. Most properties only accept values of a certain type, and within a certain range.

To see the appropriate values for the property, search Help for the property in question.

Method cannot be called in RDC's current state (Error 40507)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCMethodNotCalledC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCMethodNotCalledS"}

A method has been called that cannot be completed while an **AddNew** or **Edit** operation is in progress. For example, you cannot call the **Refresh** method while the **RemoteData control (RDC)** is editing an existing row or adding a new row. Make sure **AddNew** and **Edit** operations are completed by executing the **Update** or **CancelUpdate** method, or by using one of the *Move* methods before calling the method that caused the error.

For additional information, search Help for the method in question.

One or more of the arguments is invalid (Error 40508)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgInvalidArgC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgInvalidArgS"}

A value of at least one of the arguments called by this method is beyond the valid range of values for the argument. Use valid argument values to call the method.

For additional information, search Help for the method in question.

Out of memory (Error 40510)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgOutOfMemoryS"}

More system resources were required than are available. This error might have one or more of the following causes and solutions:

- You have too many applications, documents, or source files open.

  Close any unnecessary applications, documents, or source files.
- You have too many forms or controls loaded.

  Eliminate unnecessary overhead or break up your application so that fewer forms and controls are loaded at once.
- You have run out of space for **Public** variables.

  Reduce the number of **Public** variables.
- You have exhausted available TEMP or virtual memory space.

  Use the System Resource meter to view available system resources. Check available disk space.
- You have insufficient RAM to run the application or set of applications loaded in memory.

  Increase the amount of available RAM by installing additional memory, or reallocate memory to reduce the size of SmartDrive or other cache memory allocations.
- Your application has generated a memory leak; it allocates memory but does not free it when it is no longer needed.
- You have written reentrant code that is not properly executed or procedures that allocate excessive memory for arrays.

Property cannot be set in RDC's current state (Error 40513)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCPropertyNotSettableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCPropertyNotSettableS"}

Some properties of the **RemoteData control (RDC)** cannot be set after you have programmatically set the **Resultset** property. If you create an **rdoResultset** object in code and set it to the **Resultset** property of the **RemoteData** control, the **RemoteData** control cannot automatically reset certain properties, such as **DataSourceName**, **Options**, **Password**, **QueryTimeout**, and **UserName**. The **RemoteData** control cannot reset these properties because the **rdoResultset** object was created outside the **RemoteData** control.

To reset these properties, close the current **rdoResultset** object, programmatically set the **RemoteData** control properties to new values, and call the **Refresh** method against the **RemoteData** control to rebuild the result set.

For additional information, search Help for the property in question.

Property not available in RDC's current state (Error 40514)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCPropertyUnavailableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCPropertyUnavailableS"}

Because some properties cannot be accessed until a valid **RemoteData control (RDC)** / RDO connection is established, the state of the **RemoteData** control restricts read access for this property. Set the appropriate **RemoteData** control properties and use the **Refresh** method to establish a connection.

For additional information, search Help for the property in question.

Resultset is empty (Error 40509)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRSEmptyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRSEmptyS"}

The result set is empty, so an operation that requires the **UpdateRow** method cannot be called for a nonexistent row. Make sure the result set is not empty before calling **UpdateRow**.

Resultset not available (Error 40511)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCResultsetUnavailableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCResultsetUnavailableS"}

The **Resultset** property cannot find a valid result set because an error occurred while opening the result set, or the result set is closed. Set the **SQL** property to a valid value and/or use the **Refresh** method against the **RemoteData control**.

The connection is not open (Error 40512)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgRDCConnectionNotOpenC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRDCConnectionNotOpenS"}

Because there is no connection to a database, the **Connection** property cannot reference a valid database connection. To establish a database connection, check the **SQL** property, set the **DataSourceName** or **Connection** property to a valid value, and use the **Refresh** method against the **RemoteData control**.

Type mismatch (Error 40515)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgTypeMismatchC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgTypeMismatchS"}

The wrong argument type was passed in the **RemoteData control** event parameter. Check the argument's values.

For additional information, search Help for the event in question.

Cannot connect to Remote Data Object (Error 40516)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmsgError40516C;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgError40516S"}

The **RemoteData** control could not create a Remote Data Object. This can occur if the Microsoft Remote Data Object library was not registered correctly, or if the library is not present on the computer.

To manually register the Remote Data Object library, type the following at the command prompt:

```
regsvr32 msrdo32.dll
```

If MSRDO32.DLL is not present on your system, reinstall Visual Basic or your application and make sure you have purchased the Enterprise Edition and the application setup program properly installs it.

A control bound to a data control using this result set canceled the operation, or was unable to synchronize itself with the result set (Error 40503).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCONTROLCANCELLEDOPS"}

Either a control bound to a data control using the current result set cancelled the current operation, or it was unable to synchronize itself with the current result set.

A control canceled the operation or an unexpected internal error has occurred (Error 40503).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCONTROLINTERNALS"}

Either a control canceled the current operation, or an unexpected internal error occurred, causing the operation to cancel.

Already at BOF (Error 40030)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgALREADYATBOFS"}

You attempted to use a current row repositioning method like **MovePrevious** or **MoveFirst** when the **BOF** property is **True**.

Already at EOF (Error 40031)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgALREADYATEOFS"}

You attempted to use a current row repositioning method like **MoveNext** or **MoveLast** when the **EOF** property is **True**.

An attempt was made to issue a SELECT statement using the Execute method (Error 40057).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgEXECUTES"}

The **Execute** method is used to execute action queries which do not return rows. The query executed contains one or more SELECT statements. In some cases, stored procedures call other stored procedures that contain SELECT statements.

An error has occurred with no information available (Error 40501).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOINFOS"}

RDO has encountered an internal error. No further information is available about the cause of the error.

An error has occurred. Unable to retrieve error information (Error 40501).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgERRORINFOS"}

RDO has encountered an internal error. No further information is available about the cause of the error.

An error occurred loading the Win32 library (version.dll) (Error 40016).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgLOADVERSIONS"}

An invalid ODBC handle was encountered (Error 40004).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgODBCHANDLES"}

Your code referenced an ODBC handle such as the **hEnv**, **hDbc** or **hStmt** properties that were not valid. If you used the ODBC API directly, you might have made one or more handles invalid.

An unexpected internal error has occurred (Error 40503).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUNEXPECTEDINTERNALS"}

This type of error can be caused by a variety of internal validity checks which indicate your application might be out of resources, or have other indeterminate problems.

Can't assign value to non-updatable field (Error 40038).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgFIELDNOTUPDATEABLES"}

You attempted to change the Value property of a database column that is marked as read-only.

Can't find table to update (Error 40079).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgTABLENOTFUNDS"}

The SQL parser could not determine the table to be updated.

Can't open result set for unnamed table (Error 40020).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUNAMEDTABLES"}

You must name any tables referenced with the **rdoTable** object.

Cancel has been selected in an ODBC dialog requesting parameters needed to complete a connection (Error 40059).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgDLGCANCELEDS"}

The user chose "Cancel" when prompted for needed connection parameters such as user-id, password or data source name.

Cannot connect to RemoteData Objects (Error 40516).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNORDOS"}

This is caused by a failure to connect to the specified data source. Failure to connect can be caused by invalid network, remote server, user authorization or other options. It can also be caused by hardware failure of the network, the remote server, or any associated connectivity components.

Could not load resource library corresponding to rdoLocaleID (Error 40061)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgLOADRESLIBS"}

The locale DLL was not found or is corrupt.

Could not refresh controls (Error 40504).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCONTROLREFRESHS"}

The **RemoteData** control could not refresh the associated bound controls.

Cursor type should be: rdOpenForwardOnly. LockType should be: rdConcurReadOnly. RowsetSize should be: 1 (Error 40080).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgMUSTFORREADONES"}

For this type of operation, you must use a "cursor-less" result set. By setting the **OpenResultset** options as shown, RDO creates a cursor-less result set.

DataType Conversion Error (Error 40517).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgDATACONVS"}

This error is usually caused by an inability to convert **Variant** data types when trying to set a data column from a bound control.

Failed to load RDOCURS.DLL (Error 40078).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgFAILLOADFOXS"}

The batch cursor library DLL was not found or is corrupt.

General Client Cursor error (Error 40069).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCLIENTCURSORGENS"}

An attempt was made to execute a **BatchUpdate** or **Requery** method without an associated Query object.

Incompatible data types for compare (Error 40014).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINCOMPATIBLETYPESS"}

You attempted to compare incompatible data types. For example, you tried to compare a binary type with a numeric type.

Incorrect CursorDiver. Only rdUseClient supported (Error 40075).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCONNNOTFOXS"}

You cannot set the **ActiveConnection** property unless you are using the Client Batch cursor library.

Incorrect type found when assigning rdoParameter Value property (Error 40044).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINCORRECTPARAMTYPES"}

The data type assigned to the **rdoParameter** in question does not match the data type of the value being assigned. For example, the data type is set to Integer and you are assigning a String value.

Whenever possible, RDO chooses the parameter data types based on information passed back from the remote data source. You can override that data type by setting the **Type** property of the **rdoParameter** object to the correct type.

Invalid operation for forward-only cursor (Error 40008).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgOPFWDONLYS"}

Forward-only cursors do not support the current row positioning method being used. You can only use the **MoveNext** method to reposition a forward-only cursor.

Invalid Operation (Error 40055).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINVALIDOPS"}

This operation is invalid in this context.

Invalid Option flags (Error 40085).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINVALIDOPTIONS"}

The options chosen are invalid in this context.

Invalid property value (Error 40505).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINVALIDPROPERTYS"}

The value being assigned is invalid for this property.

Invalid row for AddNew (Error 40010).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgADDNEWNOROWS"}

The current row pointer is invalid for this operation.

Invalid state for Move (Error 40023).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINVALIDSTATES"}

You cannot use a **Move** method in this context.

Method cannot be called in RemoteData control's current state (Error 40507).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgMETHODS"}

You cannot use this method in the current **RemoteData** control state.

No current row (Error 40009).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgEDITNOROWS"}

You attempted to edit, delete or perform some other operation when the current row pointer was not positioned over a valid row. This can occur when:

- All rows of a result set are deleted.
- There are no rows in a result set.
- One or more rows' bookmarks are made invalid.

Object Collection : Can't add non-object item (Error 40053).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNONOBJECTS"}

An attempt was made to add an object to a collection which was of a different type. For example attempting to add an **rdoResultset** object to an **rdoQueries** collection would trigger this error.

Object Collection:Assignment to Count property not allowed (Error 40051).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgADDCOUNTS"}

The **Count** property of this collection cannot be set in code. It is maintained automatically by Visual Basic.

Object Collection:Illegal modification - collection is read-only (Error 40049).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCOLLECREADONLYS"}

An attempt was made to remove a member from a read-only collection.

Object is invalid or not set (Error 40011).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgCLOSEDS"}

You attempted to assign a value using an object that was not Set.

One or more of the arguments is invalid (Error 40508).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINVALIDARGUMENTS"}

The arguments supplied are not valid in this context.

Only the rdUseClientBatch cursor driver can support rdConcurBatch lock type (Error 40081).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgBATCHNOTSUPPORTS"}

The Client Batch cursor library must be used whenever working with optimistic batch updates or dissociate **rdoResultset** objects.

RemoteData control reports: Out of memory (Error 40510).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgMEMORYS"}

This can be caused by any of several internal resources being exhausted including RAM, swap space on disk, or internal handles.

Partial equality requires string column (Error 40013).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgPARTIALCMPS"}

Property cannot be set in RemoteData control's current state (Error 40513).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgSETPROPERTYS"}

Some **RemoteData** Control properties cannot be set at runtime.

Property not available in RemoteData control's current state (Error 40514).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgAVAILPROPERTYS"}

Some **RemoteData** Control properties are not available at runtime.

Result set is empty (Error 40509).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRESULTEMPTYS"}

No rows were returned for the result set or all rows have been deleted.

Result set not available (Error 40511).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRESULTUNAVAILABLES"}

The result set cannot be referenced in its current state.

The column buffer is bound so data cannot be appended. Use Value property to set data (Error 40048).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOAPPENDONBUNDS"}

The connection is not open (Error 40512).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOCONNECTIONS"}

The operation attempted requires that the **rdoConnection** object be associated with a specific server. Use the **EstablishConnection** or **OpenConnection** method to open the connection.

The object is already in the collection (Error 40077).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgALREADYINLISTS"}

You cannot add another member to this collection using the name provided. There is already another member in the collection with this name.

The object is still in some other collection (Error 40083).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINOTHERCOLLECTIONS"}

The rdoConnection object is not connected to a data source (Error 40071).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOTCONNECTEDS"}

rdoConnection objects can be instantiated but not connected to a specific data source. To perform this operation, you must first use the **EstablishConnection** or **OpenConnection** method to open the connection.

The rdoConnection object is already connected to a data source (Error 40072).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgALREADYCONNECTS"}

You attempted to use the **EstablishConnection** method against an **rdoConnection** object while it is still connected to a data source. Use the **Close** method to disconnect from the current connection before retrying the operation.

The rdoConnection object is busy connecting (Error 40073).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgINCONNECTINGS"}

You cannot use the **rdoConnection** object until the **StillConnecting** property returns False.

The rdoQuery or rdoResultset has no active connection (Error 40074).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOACTIVECONNS"}

The operation you attempted is not permitted unless the object has an active connection.

The result set is read-only (Error 40058).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgREADONLYS"}

You attempted to update, add or delete rows in a result set that does not support these operations.

This column does not have the ChunkRequired flag set. Use the Value property to retrieve its contents (Error 40060).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgNOTCHUNKS"}

You should not use the **GetChunk** method with columns that do not require its use.

This function is not supported (Error 40082).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgBATCHNOTSUPPORTDAOS"}

This operation is not supported in this context.

This property is currently read-only (Error 40076).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgPROPREADONLYS"}

This property cannot be altered in the current state of its object.

Type mismatch (Error 40515).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgTYPES"}

The data types do not match for the column or parameter specified.

You attempted to reposition to a deleted row (Error 40056).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgRECDELETEDS"}

Either you, or another user sharing this data has deleted the row in question.

You must use AppendChunk to set data in a TEXT or IMAGE (BLOB) field (Error 40052).

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmsgUSEAPPENDCHUNKS"}

The number of bytes in the field require use of the **AppendChunk** method. Check the **ChunkRequired** property to determine when the chunk methods are required.

Error Event (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtErrorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtErrorA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtErrorS"}

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

Private Sub *object* **_Error([***index* **As Integer**,]*Number* **As Long,** *Description* **As String,** *Scode* **As Long,** *Source* **As String,** *HelpFile* **As String,** *HelpContext* **As Long,** *CancelDisplay* **As Boolean)**

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
Number	The native error number.
Description	Describes the error.
Scode	ODBC error return code.
Source	Source of the error.
HelpFile	The path to a Help file containing more information on the error.
HelpContext	The Help file context number.
CancelDisplay	A number corresponding to the action you want to take, as described in Settings.

Settings

The settings for *CancelDisplay* are:

Constant	Value	Description
rdDataErrContinue	0	Continue.
rdDataErrDisplay	1	(Default) Display the error message.

Remarks

Generally, the Error event arguments correspond to the properties of the **rdoError** object.

You usually provide error-handling functionality for run-time errors in your code. However, run-time errors can occur when none of your code is running, as when:

- A user clicks a **RemoteData control** button.
- The **RemoteData** control attempts to open an **rdoConnection** and creates **rdoResultset** objects after the Form_Load event.
- A custom control performs an operation, such as the **MoveNext** method, the **AddNew** method, or the **Delete** method.

If an error results from one of these actions, the Error event occurs.

If you don't code an event procedure for the Error event, Visual Basic displays the message associated with the error.

Reposition Event (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtRepositionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtRepositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtRepositionS"}

Occurs after a row becomes the current row.

Syntax

Private Sub *object*.**Reposition (**[*index* **As Integer**]**)**

The Reposition event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

When a **RemoteData control** is loaded, the first row in the **rdoResultset** object becomes the current row, causing the Reposition event to fire. The Reposition event fires after each row becomes current:

Whenever a user clicks any button on the **RemoteData** control to move from row to row.

- You use one of the *Move* methods, such as **MoveNext**.
- You use any other property or method that changes the current row

In contrast, the Validate event occurs before moving to a different row. The RowCurrencyChange event associated with the **rdoResultset** also fires when the current result set row changes.

You can use the Reposition event to perform calculations based on data in the current row or to change the form in response to data in the current row.

Validate Event (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtValidateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtValidateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtValidateS"}

Occurs before a different row becomes the current row; before the **Update** method (except when data is saved with the **UpdateRow** method); and before a **Delete**, **Unload**, or **Close** operation.

Syntax

Private Sub *object*_**Validate (**[*index* **As Integer**,] *action* **As Integer**, *save* **As Integer)**

The Validate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
action	An **Integer** or constant that indicates the operation causing this event to occur, as described in Settings.
save	A **Boolean** expression that specifies whether bound data has changed, as described in Settings.

Settings

The settings for *action* are:

Constant	Value	Description
rdActionCancel	0	Cancel the operation when the **Sub** exits.
rdActionMoveFirst	1	**MoveFirst** method.
rdActionMovePrevious	2	**MovePrevious** method.
rdActionMoveNext	3	**MoveNext** method.
rdActionMoveLast	4	**MoveLast** method.
rdActionAddNew	5	**AddNew** method.
rdActionUpdate	6	**Update** operation (not **UpdateRow**).
rdActionDelete	7	**Delete** method.
rdActionFind	8	**Find** method (not implemented).
rdActionBookmark	9	The **Bookmark** property has been set.
rdActionClose	10	The **Close** method.
rdActionUnload	11	The form is being unloaded.
rdActionUpdateAddNew	12	A new row was inserted into the result set.
rdActionUpdateModified	13	The current row changed.
rdActionRefresh	14	**Refresh** method executed.
rdActionCancelUpdate	15	Update canceled.
rdActionBeginTransact	16	**BeginTrans** method.
rdActionCommitTrans	17	**CommitTrans** Method.

act		
rdActionRollbackTransact	18	**RollbackTrans** Method
rdActionNewParameters	19	Change in parameters, or order of columns or rows.
rdActionNewSQL	20	SQL statement changed.

The settings for *save* are:

Setting	Description
True	A **Boolean** expression indicating bound data has changed.
False	A **Boolean** expression indicating bound data has not changed.

Remarks

The ***save*** argument initially indicates whether bound data has changed. This argument can still be **False** if data in the copy buffer is changed. If ***save*** is **True** when this event exits, the **Edit** and **UpdateRow** methods are invoked.

This event can occur regardless of whether data in bound controls changes, or whether bound controls exist. You can use this event to change values and update data. You can also choose to save data or stop whatever action is causing the event to occur and substitute a different action.

You can change the various *Move* methods and the **AddNew** method, which can be freely exchanged (any *Move* into **AddNew**, any *Move* into any other *Move*, or **AddNew** into any *Move*). Attempting to change **AddNew** or one of the *Moves* into any of the other actions is either ignored or produces a trappable error. Any action can be stopped by setting ***action*** to **rdActionCancel**. If you change the ***action*** argument, the current action will also be canceled.

In your code for this event, you can check the data in each bound control where **DataChanged** is **True**. You can then set **DataChanged** to **False** to avoid saving that data in the database.

Note Because a data-aware control can have more than one bound property, the **DataChanged** property must be examined for each of the bound properties as enumerated in the **Bindings** collection.

You can't use any methods (such as **MoveNext**) on the underlying **rdoResultset** object during this event.

QueryCompleted Event (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtQueryCompletedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtQueryCompletedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtQueryCompletedS"}

Occurs after the query of an **rdoResultset** generated by a **RemoteData** Control returns the first result set.

Syntax

Private Sub *object*.**QueryCompleted ([***index* **As Integer])**

The QueryCompleted event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.

Remarks

When a **RemoteData control** completes the creation of an **rdoResultset**, the QueryCompleted event is fired. This event is not triggered if you execute the **Cancel** method which terminates processing of the query before the query has been completed.

This event fires for both asynchronous and synchronous query operations.

RDO Events Example

This example illustrates several of the Remote Data Object (RDO) event handlers. The code establishes event variables and handlers to trap connection and query events. To help illustrate use of the BeforeConnect event, the code concatenates a workstation ID value and the current time to the end of the connect string. This permits identification of the specific connection at the server. After establishing the connection, the code executes a query that takes an fairly long time to execute — the query is designed to run for about a minute. Because a 5 second QueryTimeout value is set, the QueryTimeout event should fire unless the query returns before 5 seconds has elapsed. Notice that the query itself is run asynchronously and the code does not poll for completion of the query. In this case the code simply waits for the QueryComplete or QueryTimeout events to fire — indicating that the query is finished. The code also permits you to request another 5 seconds of waiting time.

```
Option Explicit
Private WithEvents cn As rdoConnection
Private WithEvents EngEv As rdoEngine
Dim er As rdoError
Dim strConnect As String
Dim rs As rdoResultset
Dim TimeStart As Single
Dim clock As Integer

Private Sub EngEv_InfoMessage()
    InfoMsg = "For your information..." _
   & " the following message" _
        & " was returned by the server." & vbCrLf
    For Each er In rdoErrors
        InfoMsg = InfoMsg & er.Number _
   & " - " & er.Description & vbCrLf
    Next

End Sub
Private Sub cn_BeforeConnect( _
   ConnectString As String, Prompt As Variant)
  InfoMsg = "About to connect to:" & ConnectString  _
   & " - " & Prompt
  ConnectString = ConnectString & ";WSID="  _
   & "EventTest" & Time$ & ";"
End Sub

Private Sub cn_Connect()    'Fires once connected.
  Connected = True
End Sub

Private Sub cn_Disconnect()'Fires when disconnected
  Connected = False
End Sub

Private Sub cn_QueryComplete( _
   ByVal ErrorOccured As Boolean)
  Timer1.Enabled = False
  QueryComplete = vbChecked
  RunButton.Enabled = True
  Beep

  MsgBox "Query Done"
```

```
End Sub

Private Sub cn_QueryTimeout(Cancel As Boolean)
    ans = MsgBox("The query did not complete "
   & "in the time allocated. " _
   & "Press Cancel to abandon the query " _
   & "or Retry to keep working.", _
        vbRetryCancel + vbQuestion, "Query Timed Out")
    If ans = vbRetry Then
        Cancel = False
        QueryComplete = vbGrayed
    Else
        Timer1.Enabled = False
        QueryComplete = vbChecked
    End If
End Sub

Private Sub MenufileExit_Click()
cn.Close
Unload Form1
End Sub

Private Sub RunButton_Click()
    RunButton.Enabled = False
    On Error GoTo C1EH
    QueryComplete = vbGrayed
    Timer1.Enabled = True
    Set rs = cn.OpenResultset( _
      "execute VeryLongProcedure", _
        rdOpenKeyset, rdConcurValues, rdAsyncEnable)
    TimeStart = Timer
QuitRun:
Exit Sub
C1EH:
    Debug.Print Err, Error
        InfoMsg = "Error:.. the following error" _
        & " was returned by the server." & vbCrLf
    For Each er In rdoErrors
        InfoMsg = InfoMsg & er.Number  _
      & " - " & er.Description & vbCrLf
    Next
    MsgBox "Query Failed to run"
    Timer1.Enabled = False
    Resume QuitRun

End Sub

Private Sub Form_Load()
On Error GoTo FLeh
Set EngEv = rdoEngine
Set cn = New rdoConnection
Show
    With cn
        .Connect = "UID=;PWD=;database=Workdb;" _
            & "Server=SEQUEL;" _
            & "driver={SQL Server};DSN='';"
        .QueryTimeout = 5
```

```
        .CursorDriver = rdUseClientBatch
        .EstablishConnection rdDriverNoPrompt
    End With
Exit Sub

FLeh:
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description
    Next
    Stop
    Resume

End Sub

Private Sub Timer1_Timer()
Static ot As Integer
' Display number of seconds
ShowClock = Int(Timer - TimeStart)
If ShowClock = ot Then ShowClock.Refresh
End Sub
```

BeforeConnect Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtBeforeConnectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtBeforeConnectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtBeforeConnectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtBeforeConnectS"}

Occurs just before RDO calls the ODBC API **SQLDriverConnect** function to establish a connection to the server.

Syntax

Private Sub *object*.**BeforeConnect**(*ConnectString* as **String**, *Prompt* as **Variant**)

The BeforeConnect event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
ConnectString	A **Variant** expression that evaluates to a connect string used to provide connect parameters for the ODBC **SQLDriverConnect** function.
Prompt	Determines how the user should be prompted.

Remarks

The BeforeConnect event is fired just before RDO calls the ODBC API **SQLDriverConnect** function to establish a connection to the server. This event gives your code an opportunity to provide custom prompting, or just provide or capture connection information.

The ***ConnectString*** parameter is the ODBC connect string RDO will pass to the ODBC API **SQLDriverConnect** function. This string can be changed during this event, and RDO will use the changed value. For example, your code can provide additional parameters, or change existing parameters of the connect string.

The ***Prompt*** parameter is the ODBC prompting constant (see the **Prompt** property). This parameter will default to the value of the ***Prompt*** parameter passed in the **OpenConnection** or **EstablishConnection** methods. The developer may change this value, and RDO will use the new value when calling **SQLDriverConnect**.

Associate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtAssociateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtAssociateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtAssociateS"}

Fired after a new connection is associated with the object.

Syntax

Private Sub *object*.**Associate**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after the result set is associated with a new **rdoConnection** object. You can use this event to initialize the new connection. The **ActiveConnection** property of the associated **rdoResultset** object refers to the new connection.

For example, you can use the Associate event procedure to send a special query each time a connection is established, but before other operations are executed.

BeginTrans Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtBeginTransC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtBeginTransA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtBeginTransS"}

Occurs after the **BeginTrans** method has completed.

Syntax

Private Sub *object*.**BeginTrans**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The BeginTrans event is raised after a **BeginTrans** method has completed. This event procedure can synchronize some other process with the transaction.

CommitTrans Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtCommitTransC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtCommitTransA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtCommitTransS"}

Occurs after the **CommitTrans** method has completed.

Syntax

Private Sub *object*.**CommitTrans**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after a **CommitTrans** method has been executed. The developer can respond to this event to synchronize some other process with the transaction.

Connect Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtConnectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtConnectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtConnectS"}

Occurs after a connection is established to the server.

Private Sub *object*.**Connect**(*ErrorOccurred* **As Boolean**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
ErrorOccurred	A **Boolean** expression that determines whether the connection was successful, as described in Settings.

Settings

The *ErrorOccurred* argument will be set to one of the following values:

Value	Description
True	The connection failed.
False	The connection succeeded.

Remarks

You can catch the Connect event and do any kind of initial queries required on a new connection, such as verifying the version of the database against the version of the client or setting a default database not established in the connect string. You can also check for errors or messages returned during the process of opening the connection — or perhaps simply clear the **rdoErrors** collection of informational messages.

DataChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtDataChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtDataChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtDataChangedS"}

Occurs when the value of the column has changed.

Syntax

Private Sub *object*.**DataChanged**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after the data in a column has been changed. The new data can be accessed through the **rdoColumn** object's **Value** property. You can also use the WillChange event to prevent or modify the change about to be made on a column-by-column basis. However, once the DataChanged event fires, the change has already been committed to the database.

Disconnect Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtDisconnectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtDisconnectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtDisconnectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtDisconnectS"}

Occurs after a connection has been closed.

Private Sub *object*.**Disconnect**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Fired after a physical connection is closed. The developer can catch this event to do any clean-up work necessary.

Applies to **rdoConnection** object.

Dissociate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtDissociateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtDissociateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtDissociateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtDissociateS"}

Occurs after an **rdoResultset** object has been dissociated from a connection.

Private Sub *object*.**Dissociate**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

This event is raised after the **ActiveConnection** property has been set to **Nothing** and the result set has been dissociated from its connection.

InfoMessage Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtInfoMessageC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtInfoMessageX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtInfoMessageA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtInfoMessageS"}

Occurs when informational messages are added to the **rdoErrors** collection.

Private Sub *object*.**InfoMessage**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after RDO receives a SQL_SUCCESS_WITH_INFO return code from the ODBC Driver Manager, and populates the **rdoErrors** collection with the informational messages.

The InfoMessage event is raised once for each *set* of informational messages. Thus, if an RDO method generates several informational messages, this event is raised only once — after the last message has been added to the collection. You can trap this event and examine the contents of the **rdoErrors** collection and decide what action is appropriate.

QueryComplete Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtQueryCompleteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtQueryCompleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtQueryCompleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtQueryCompleteS"}

Occurs after the query of an **rdoResultset** returns the first result set

Syntax

Private Sub *object*.**QueryComplete**(*Query* as **rdoQuery**, *ErrorOccured* as **Boolean**)

The QueryComplete event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Query	An object expression that evaluates to an **rdoQuery** object whose query has just completed.
ErrorOccured	A **Boolean** expression indicating if an error occurred while processing the query.

The settings for ***ErrorOccurred*** are:

Setting	Description
True	An error occurred during query processing.
False	An error did not occur during query processing.

Fired when a query has completed. You can use this event as a notification that the result set is now ready for processing.

The ***ErrorOccured*** parameter indicates if there was an error while the query was executing. If this flag is **True**, you should check the **rdoErrors** collection for more information.

The QueryComplete event fires for all queries execute on this **rdoConnection**. This includes those queries executed via the **OpenResultset** or **Execute** methods, as well as those executed from an associated **rdoQuery** object. The ***Query*** argument is an object reference indicating which query just finished executing. Using this argument, you can write a single event handler for all queries on the connection, but still customize the handler for specific queries. When executing queries against the **rdoConnection** object itself, RDO creates an **rdoQuery** object internally, and a reference to this internal **rdoQuery** is passed as the ***Query*** argument.

This event should be used instead of polling the **StillExecuting** property to test for completion of **OpenResultset** or **Execute** method queries.

QueryTimeout Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtQueryTimeoutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtQueryTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtQueryTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtQueryTimeoutS"}

Occurs when the query execution time has exceeded the value set in the **QueryTimeout** property.

Syntax

Private Sub *object*.**QueryTimeout**(*Query* as **rdoQuery**, *Cancel* as **Boolean**)

The BeforeConnect event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Query	An object expression that evaluates to an **rdoQuery** object whose query has just completed.
Cancel	A **Boolean** expression indicating if an error occurred while processing the query.

The settings for ***Cancel*** are:

Setting	Description
True	RDO should cancel further query processing.
False	RDO should continue processing the query for another query timeout period.

Remarks

Fired when a running query has exceeded the time specified by the **QueryTimeout** property. This event is fired each time the **QueryTimeout** time has been reached. This event is fired on both asynchronous and synchronous queries.

The ***Cancel*** parameter indicates if RDO should cancel the query or continue processing the query and wait for the number of seconds specified in the **QueryTimeout** property. The default value of this parameter is **True**, so if your code not respond to this event, the query is canceled after the **QueryTimeout** time has been reached. If the value of the parameter is set to **False**, RDO continues to wait for the query to complete for another **QueryTimeout** period.

You can use this method to display a message box to the user asking them if they wanted to cancel the query, or continue to wait another *N* seconds.

The QueryTimeout event fires for all queries execute on this **rdoConnection**. This includes those queries executed via the **OpenResultset** or **Execute** methods, as well as those executed from an associated **rdoQuery** object. The ***Query*** argument is an object reference indicating which query just timed out. Using this argument, you can write a single event handler for all queries on the connection, but still customize the handler for specific queries. When executing queries against the **rdoConnection** object itself, RDO creates an **rdoQuery** object internally, and a reference to this internal **rdoQuery** is passed as the ***Query*** argument.

ResultsChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtResultsChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtResultsChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtResultsChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtResultsChangedS"}

Occurs when a new result set is made available after the **MoreResults** method is executed.

Private Sub *object*.**ResultsChanged**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after the **MoreResults** method completes and a new set of rows is loaded into the result set. This event is fired even if there are no more sets and the **MoreResults** method returns **False**. In this case, both the EOF and BOF properties will be **True**, indicating that the result set is empty.

RollbackTrans Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtRollbackTransC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtRollbackTransX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtRollbackTransA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtRollbackTransS"}

Occurs after the **RollbackTrans** method has completed.

Private Sub *object*.**RollbackTrans**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

You can respond to this event to synchronize some other process with the transaction.

RowCurrencyChange Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtRowCurrencyChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtRowCurrencyChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtRowCurrencyChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtRowCurrencyChangeS"}

Occurs after the result set has repositioned to a new row, BOF or EOF.

Private Sub *object*.**RowCurrencyChange**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after the result set has repositioned to a new row, or has moved to either BOF or EOF. Any of the *Move* methods, the **AbsolutePosition**, **PercentPosition**, or **Bookmark** properties, or the **Requery**, **MoreResults**, or **Update** (after an **AddNew**) methods can also cause a the current row pointer to be repositioned and cause the RowCurrencyChange event to fire. The current position can be determined by accessing the **AbsolutePosition**, **PercentPosition**, or **Bookmark** properties of the object.

The RowCurrencyChange event can be used to execute a detail query when an associated master row currency changes. For example, if you setup a form containing a master customer record, and a set of rows corresponding to customer orders, you can use the RowCurrencyChange event to launch a query that returns all associated order information each time the user chooses another master customer record.

Note The order in which the RowCurrencyChange and Reposition events fire cannot be predicted.

RowStatusChanged Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtRowStatusChangeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtRowStatusChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtRowStatusChangeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtRowStatusChangeS"}

Occurs after the data state of the current row changes due to an edit, delete or insert.

Private Sub *object*.**RowStatusChange**( )

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

This event is raised after the status of the current row data changes. The status of a row can change due to an **Delete**, or **Update** operation. The current status for the row can be determined using the **Status** property of the object.

WillAssociate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtWillAssociateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtWillAssociateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtWillAssociateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtWillAssociateS"}

Occurs before a new connection is associated with the object.

Private Sub *object*.**WillAssociate**(*Connection* as **rdoConnection,** *Cancel* as **Boolean**)

The WillAssociate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Connection	An object expression that evaluates to the **rdoConnection** object that is to be associated.
Cancel	A **Boolean** expression indicating if RDO should prohibit the association.

The settings for ***Cancel*** are:

Setting	Description
True	RDO will prohibit the association.
False	(Default) RDO will not prohibit the association.

Remarks

This event is raised after you set the **ActiveConnection** property to a valid **rdoConnection** object, but before the actual associate is made.

The ***Connection*** argument is a reference to the **rdoConnection** object that you are attempting to associate with the **rdoResultset** object. When the WillAssociate event is raised, the **ActiveConnection** property remains set to the value *before* the attempted association. You can use this property to determine the current **rdoResultset** connection association.

You can prohibit the association by setting the ***Cancel*** argument to **True**, causing RDO to not associate the result set with the new connection and produce a runtime error. If you do not prohibit the association using the ***Cancel*** argument, the **ActiveConnection** property is set to the reference contained in the ***Connection*** parameter after this event procedure completes.

WillChangeData Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtWillChangeDataC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtWillChangeDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtWillChangeDataA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtWillChangeDataS"}

Occurs before data is changed in the column.

Private Sub *object*.**WillChangeData**(*NewValue* as **Variant,** *Cancel* as **Boolean**)

The WillChangeData event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
NewValue	A **Variant** expression containing the data to be applied to the column.
Cancel	A **Boolean** expression indicating if RDO should prohibit the change.

The settings for ***Cancel*** are:

Setting	Description
True	RDO will prohibit the change.
False	(Default) RDO will not prohibit the change.

Remarks

This event is raised just before RDO commits any change to the data in a column. By trapping this event, you can either modify the new value, or prohibit the change by modifying the ***Cancel*** argument.

If you modify the ***NewValue*** parameter, the modified value is assigned to the column's **Value** property. This allows you translate or substitute data.

By default, the ***Cancel*** argument is **False**, but if you set it to **True**, the change to the column's data is canceled, and RDO generates a trappable error.

WillDissociate Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtWillDissociateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtWillDissociateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtWillDissociateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtWillDissociateS"}

Occurs before the connection is set to nothing.

Private Sub *object*.**WillDissociate**(*Cancel* as **Boolean**)

The WillDissociate event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Cancel	A **Boolean** expression indicating whether RDO should prohibit the disassociation.

The settings for ***Cancel*** are:

Setting	Description
True	RDO will prohibit the change.
False	(Default) RDO will not prohibit the change.

Remarks

This event is raised when the developer attempts to set the **ActiveConnection** property to **Nothing** but *before* the result set is dissociated from the connection.

If you wish to prohibit the dissociation, set the ***Cancel*** parameter to **True**, causing RDO to cancel the operation and trigger a trappable error.

The default value for the ***Cancel*** parameter is **False**, so if the event is not trapped, the dissociation is completed.

WillExecute Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtWillExecuteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtWillExecuteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtWillExecuteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtWillExecuteS"}

Occurs before the execution of a query,

Private Sub *object*.**WillExecute**(*Query* as **rdoQuery**, *Cancel* as **Boolean**)

The WillExecute event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Query	An object expression that evaluates to an **rdoQuery** object whose query has just completed.
Cancel	A **Boolean** expression indicating if RDO should prohibit the change.

The settings for ***Cancel*** are:

Setting	Description
True	RDO will prohibit the change.
False	(Default) RDO will not prohibit the change.

Remarks

This event is fired before the execution of a query, regardless if it is an action or row-returning query. You can trap this event to disallow the execution of certain queries, or to make last-minute adjustments to the **rdoQuery** object's SQL string.

The ***Cancel*** argument allows you to disallow the query. The ***Cancel*** parameter will default to **False**, but if you set it to **True**, the query will not execute, and RDO generates a trappable error indicating that the query was canceled.

For example, you can pre-screen the query to make sure the WHERE clause will not cause a table-scan operation. Thus, by setting the ***Cancel*** argument to **True**, you can prohibit users from searching for customers with the last name of “Smith” without also providing a first name or street address.

The WillExecute event fires for all queries execute on this **rdoConnection**. This includes those queries executed via the **OpenResultset** or **Execute** methods, as well as those executed from an associated **rdoQuery** object. The ***Query*** argument is an object reference indicating which query is about to execute. Using this argument, you can write a single event handler for all queries on the connection, but still customize the handler for specific queries. When executing queries against the **rdoConnection** object itself, RDO creates an **rdoQuery** object internally, and a reference to this internal **rdoQuery** is passed as the ***Query*** argument.

WillUpdateRows Event

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdevtWillUpdateRowsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdevtWillUpdateRowsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdevtWillUpdateRowsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdevtWillUpdateRowsS"}

Occurs before an update to the database occurs.

Private Sub *object*.**WillUpdateRows**(*ReturnCode* as **Long**)

The WillUpdateRows event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
RetrunCode	An **Long** integer expression or constant indicating whether the developer handled the update or not as described in Settings.

The ***Value*** argument is used to notify RDO about what your code did during the event handling. The possible values for this argument are as follows:

Setting	Description
rdUpdateSuccessful	Your code handled the update and was successful in doing so.
rdUpdateWithCollisions	Your code handled the update successfully, but some rows produced collisions (batch mode only).
rdUpdateFailed	Your code attempted to handle the update, but encountered an error when doing so.
rdUpdateNotHandled	Your code did not handle the update. RDO should continue notifying and if no one handles the update, RDO should update the data itself.

Remarks

The WillUpdateRows event is raised before updated, new and deleted rows are committed to the server. You can override the update behavior of the cursor by responding to this event and perform your own updates using stored procedures or any other mechanism you choose.

If the result set is using batch optimistic concurrency, this event is only raised when the **BatchUpdate** method is called. In this case, the entire set of changes is about to be transmitted to the server.

If the result set is not in a batch mode, the WillUpdateRows event is raised for each call to the **Update** method, since the changes for that row are immediately sent to the server.

To summarize, no matter what mode the result set is in, this event is only raised before data is actually sent to the server.

If you set the ***ReturnCode*** argument to **rdUpdateSuccessful**, RDO assumes that your code successfully handled the update. RDO will not send this event to any additional clients (if there is more than one handler of this event) and the status for the row(s) and their columns is set to **rdRowUnmodified** and **rdColUnmodified** respectively.

If you set the ***ReturnCode*** parameter to **rdUpdateWithCollisions**, RDO assumes that you have successfully handled the update, but some rows caused collisions. RDO will not send this event to any additional clients (if there was more than one handler of this event) and the status for the rows and their columns is not changed. It is your code's responsibility to set the column status flags during the handling of this event. The **rdUpdateWithCollisions** would only be used if you are using batch

optimistic concurrency and you wanted to check for and handle collisions in code.

If the developer sets the ***ReturnCode*** parameter to **rdUpdateFailed**, RDO assumes that your code attempted to handle the update, but encountered an error while doing so. RDO will not send this event to any additional clients (if there was more than one handler of this event) and the status for the row(s) and their columns remains unchanged. Finally, RDO generates a runtime error to be trapped by the **Update** method causing the WillUpdate event to fire.

If you set the ***ReturnCode*** parameter to **rdUpdateNotHandled**, RDO will assume that the developer did not handle the update, and RDO will continue to raise this event to all remaining clients (if there was more than one handler of this event). If all clients return rdUpdateNotHandled, RDO will perform the update itself, according the normal rules.

The default value for the ***ReturnCode*** parameter is **rdUpdateNotHandled**, so if no client sinks the event, or no client changes the value of ***ReturnCode***, RDO will perform the update.

AddNew Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthAddNewC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthAddNewX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthAddNewA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthAddNewS"}

Creates a new row for an updatable **rdoResultset** object.

Syntax

object.**AddNew**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **AddNew** method prepares a new row you can edit and subsequently add to the **rdoResultset** object named by *object* using the **Update** method. This method initializes the columns to SQL_IGNORE to ensure columns not specifically referenced are not included in the update operation.

When the **AddNew** method is executed, the **EditMode** property is set to **rdEditAdd** until you execute the **Update** method.

After you modify the new row, use the **Update** method to save the changes and add the row to the result set. No changes are made to the database until you use the **Update** method — unless you are using the Client Batch cursor library — which does not write to the database until the **BatchUpdate** method is used.

The **AddNew** method does not return an error if the **rdoResultset** is not updatable. A trappable error is triggered when the **Update** method is used against an object that is not updatable. For an object to be updatable, the **rdoColumn**, **rdoResultset**, and **rdoConnection** objects must all be updatable — check the **Updatable** property of each of these objects before performing an update. There are a variety of reasons why an **rdoResultset** is not updatable as discussed in the **Update** method topic.

Caution If you use the **AddNew** method on a row and then perform any operation that moves to another row without using **Update**, your changes are lost without warning. In addition, if you close the *object* or end the procedure which declares the *object* or its **rdoConnection** object, the new row and the changes made to it are discarded without warning.

A newly added row might be visible as a part of the **rdoResultset** if your data source and type of cursor support it. For example, newly added rows are not included in a static-type **rdoResultset**.

When newly added rows are included in the **rdoResultset**, the row that was current *before* you used **AddNew** remains current. When the row is added to the cursor keyset, and you want to make the new row current, you can set the **Bookmark** property to the bookmark identified by the **LastModified** property setting.

If you need to cancel a pending **AddNew** operation, use the **CancelUpdate** method.

When you use the **Update** method after using the **AddNew** method, the RowCurrencyChange event is fired.

AddNew, Update, CancelUpdate Method Example

The following example illustrates use of the **AddNew** method to add new rows to a base table. This example assumes that you have read-write access to the table, that the column data provided meets the rules and other constraints associated with the table, and there is a unique index on the table. The data values for the operation are taken from three **TextBox** controls on the form. Note that the unique key for this table is not provided here as it is provided automatically – it is an *identity* column.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim col As rdoColumn

Private Sub AddNewJob_Click()
On Error GoTo ANEH

With rs
   .AddNew
   !job_desc = JobDescription
   !min_lvl = MinLevel
   !max_lvl = MaxLevel
   .Update
End With
Exit Sub

UpdateFailed:
MsgBox "Update did not suceed."
rs.CancelUpdate
Exit Sub
A
NEH:
Debug.Print Err, Error
For Each er In rdoErrors
   Debug.Print er
Next
Resume UpdateFailed

End Sub

Private Sub Form_Load()

cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=sequel;" _
   & "driver={SQL Server};database=pubs;dsn='';"
cn.EstablishConnection
With qy
   .Name = "JobsQuery"
   .SQL = "Select * from Jobs"
   .RowsetSize = 1
   Set .ActiveConnection = cn
   Set rs = .OpenResultset(rdOpenKeyset, _
      rdConcurRowver)
   Debug.Print rs.Updatable
End With
```

```
Exit Sub
End Sub
```

AppendChunk Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthAppendChunkC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthAppendChunkX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthAppendChunkA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthAppendChunkS"}

Appends data from a **Variant** expression to an **rdoColumn** object with a data type of **rdTypeLONGVARBINARY** or **rdTypeLONGVARCHAR**.

Syntax

object **!** *column***.AppendChunk** *source*

The **AppendChunk** method syntax has these parts:

Part	Description
object	An object expression that evaluates to the **rdoResultset** object containing the **rdoColumns** collection.
column	An object expression that evaluates to an **rdoColumn** object whose **ChunkRequired** property is set to **True**.
source	A string expression or variable containing the data you want to append to the **rdoColumn** object specified by *column.*

Remarks

Chunk data columns are designed to store large binary (BLOB) or text values that can range in size from a few characters to over 1.2GB and are stored in the database on successive data pages. In many cases, *chunk* data cannot be managed with a single operation, so you must use the *chunk* methods to save and write data. If the **ChunkRequired** property is **True** for a column, you should use the **AppendChunk** method to manipulate column data. However, if there is sufficient internal memory available, RDO might be able to carry out the operation without use of the **AppendChunk** method. In other words, you might be able to simply assign a value to a BLOB column.

Use the **AppendChunk** method to write successive blocks of data to the database column and **GetChunk** to extract data from the database column. Certain operations (copying, for example) involve temporary strings. If string space is limited, you may need to work with smaller segments of a *chunk* column instead of the entire column.

Use the **BindThreshold** property to specify the largest column size that will be automatically bound.

Use the **ColumnSize** property to determine the number of bytes in a *chunk* column. Note that for variable-sized columns, it is not necessary to write back the same number of bytes as returned by the **ColumnSize** property as **ColumnSize** reflects the size of the column before changes are made.

If there is no current row when you use **AppendChunk**, a trappable error occurs.

Note The initial **AppendChunk** (after the first **Edit** method), even if the row already contains data, replaces existing column data. Subsequent **AppendChunk** calls within a single **Edit** session appends data to existing column data.

AppendChunk, GetChunk Method Example

This example illustrates use of the **AppendChunk** and **GetChunk** methods to write page-based binary large object (BLOB) data to a remote data source. The code expects a table with a char, text, and image field named *Chunks*. To create this table, submit the following as an action query against your test database:

```
CREATE TABLE Chunks (ID integer identity NOT NULL, PName char(10) NULL,
Description TEXT NULL,
Photo IMAGE NULL)
CREATE UNIQUE INDEX ChunkIDIndex on Chunks(ID)
```

Once the table is created, you will need to locate one or more .BMP or other suitable graphics images that can be loaded by the **PictureBox** control.

```
'
Option Explicit
Dim en As rdoEnvironment
Dim Qd As rdoQuery
Dim Cn As rdoConnection
Dim Rs As rdoResultset
Dim SQL As String
Dim DataFile As Integer, Fl As Long, Chunks As Integer
Dim Fragment As Integer, Chunk() As Byte, I As Integer
Const ChunkSize As Integer = 16384

Private Sub Form_Load()
Set en = rdoEnvironments(0)
Set Cn = en.OpenConnection(dsname:="", _
Connect:="UID=;PWD=;DATABASE=WorkDB;" _
& "Driver={SQL Server};SERVER=Betav486", _
prompt:=rdDriverNoPrompt)
Set Qd = Cn.CreateQuery("TestChunk", "Select * from Chunks Where PName
= ?")
End Sub
Private Sub LoadFromFile_Click()
'
'  Locates a file and sets the Filename to this file.
'
With CommonDialog1
.Filter = "Pictures(*.bmp;*.ico)|*.bmp;*.ico"
.ShowOpen
FileName = .FileName
End With
End Sub

Private Sub ReadFromDB_Click()
If Len(NameWanted) = 0 Then _
NameWanted = InputBox("Enter name wanted", "Animal")
Qd(0) = NameWanted
Set Rs = Qd.OpenResultset(rdOpenKeyset, rdConcurRowver)
If Rs Is Nothing Or Rs.Updatable = False Then
MsgBox "Can't open or write to result set"
Exit Sub
End If
If Rs.EOF Then
```

```
MsgBox "Can't find picture by that name"
Exit Sub
End If
Description = Rs!Description
DataFile = 1
Open "pictemp" For Binary Access Write As DataFile
Fl = Rs!Photo.ColumnSize
Chunks = Fl \ ChunkSize
Fragment = Fl Mod ChunkSize
ReDim Chunk(Fragment)
Chunk() = Rs!Photo.GetChunk(Fragment)
Put DataFile, , Chunk()
For I = 1 To Chunks
ReDim Buffer(ChunkSize)
Chunk() = Rs!Photo.GetChunk(ChunkSize)
Put DataFile, , Chunk()
Next I
Close DataFile
FileName = "pictemp"
End Sub

Private Sub SaveToDB_Click()
If Len(NameWanted) = 0 Then _
NameWanted = InputBox("Enter name for this" _
& " picture", "Animal")
Qd(0) = NameWanted
Set Rs = Qd.OpenResultset(rdOpenKeyset, _
rdConcurRowver)
If Rs Is Nothing Or Rs.Updatable = False Then
MsgBox "Can't open or write to result set"
Exit Sub
End If
If Rs.EOF Then
Rs.AddNew
Rs!PName = NameWanted
If Description = "" Then  _
Description = InputBox("Describe the picture", _
"Don't care")
'Rs!Description = Description
Else
Rs.Edit
End If
DataFile = 1
Open FileName For Binary Access Read As DataFile
Fl = LOF(DataFile)    ' Length of data in file
If Fl = 0 Then Close DataFile: Exit Sub
Chunks = Fl \ ChunkSize
Fragment = Fl Mod ChunkSize
Rs!Photo.AppendChunk Null
ReDim Chunk(Fragment)
Get DataFile, , Chunk()
Rs!Photo.AppendChunk Chunk()
ReDim Chunk(ChunkSize)
For I = 1 To Chunks
Get DataFile, , Chunk()
Rs!Photo.AppendChunk Chunk()
Next I
```

```
Close DataFile
Rs.Update
End Sub

Private Sub FileName_Change()
Picture1.Picture = LoadPicture(FileName)
End Sub
```

BeginTrans, CommitTrans, RollbackTrans Methods (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthBeginTransC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthBeginTransA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthBeginTransS"}

The transaction methods manage transaction processing during a session represented by the *object* placeholder as follows:

- **BeginTrans** begins a new transaction.
- **CommitTrans** ends the current transaction and saves the changes.
- **RollbackTrans** ends the current transaction and restores the databases in the **rdoEnvironment** object to the state they were in when the current transaction began.

You can use the transaction methods with an **rdoConnection** object — but in this case, the transaction scope only includes **rdoResultset** and **rdoQuery** objects created under the **rdoConnection**.

Syntax

object.**BeginTrans | CommitTrans | RollbackTrans**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

You use the transaction methods with an **rdoEnvironment** or **rdoConnection** object when you want to treat a series of changes made to the databases in a session as one logical unit. That is, either the set of operations completes as a set, or is rolled back as a set. This way if any operation in the set fails, the entire transaction fails. Transactions also permit you to make *temporary* changes to the database – changes that can be undone with the **RollbackTrans** method.

Typically, ODBC drivers work in one of two modes:

- **Auto-commit Mode:** When you have not explicitly started a transaction using the BeginTrans method, every operation executed is immediately committed to the database upon completion.
- **Manual-commit Mode:** When you explicitly start a transaction using the **BeginTrans** method or use the ODBC **SQLSetStmtOption** function to disable the SQL_AUTO_COMMIT mode, or send an SQL statement to begin a transaction (BEGIN TRANS), operations are part of a transaction and no changes are committed to the database until you use the **CommitTrans** method. If the connection fails before **CommitTrans** is executed, or you use the **RollbackTrans** method, the operations are undone — rolled back.

Note When working with remote servers that support a Distributed Transaction Coordinator (DTC) like Microsoft SQL Server, you can initiate and control transactions that span more than one server. That is, if you invoke a procedure on the remote server that invokes a remote procedure call, the DTC service can ensure that this operation is included in the initial transaction. See *Building Client/Server Applications with Visual Basic* for more information.

Typically, you use transactions to maintain the integrity of your data when you must update rows in two or more tables and ensure that changes made are completed (committed) in all tables or none at all (rolled back). For example, if you transfer money from one account to another, you might subtract an amount from one and add the amount to another. If either update fails, the accounts no longer balance. Use the **BeginTrans** method before updating the first row, and then, if any subsequent update fails, you can use the **RollbackTrans** method to undo all of the updates. Use the **CommitTrans** method after you successfully update the last row.

Caution Within one **rdoEnvironment** object, transactions are always global to the **rdoEnvironment** and aren't limited to only one database or result set. If you perform operations on

more than one database or result set within an **rdoEnvironment** transaction, the **RollbackTrans** method restores all operations on those databases and result sets.

Once you use **CommitTrans**, you can't undo changes made during that transaction unless the transaction is nested within another transaction that is itself rolled back. You cannot nest transactions unless you use an action query to directly execute SQL transaction management statements. If you want to have simultaneous transactions with overlapping, non-nested scopes, you can create additional **rdoEnvironment** objects to contain the concurrent transactions.

Note You can use SQL action queries that contain transaction statements. For example, with Microsoft SQL Server, you can use SQL statements like BEGIN TRANSACTION, COMMIT TRANSACTION, or ROLLBACK TRANSACTION. This technique supports nested transactions which may not be supported by the ODBC driver.

If you close an **rdoEnvironment** object without saving or rolling back any pending transactions, the transactions are automatically rolled back.

No error occurs If you use the **CommitTrans** or **RollbackTrans** method without first using the **BeginTrans** method.

Some databases may not support transactions, in which case the **Transactions** property of the **rdoConnection** object or **rdoResultset** object is **False**. To make sure that the database supports transactions, check the value of the **Transactions** property of the **rdoConnection** object before using the **BeginTrans** method. If you are using an **rdoResultset** object based on more than one database, check the **Transactions** property of the **rdoResultset** object. If the **rdoConnection** or **rdoResultset** doesn't support transactions, the methods are ignored and no error occurs.

Cancel Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCancelC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCancelA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCancelS"}

Cancels the processing of a query running in asynchronous mode, or cancels any pending results against the specified RDO object.

Syntax

object.**Cancel**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **Cancel** method *requests* that the remote data source stop work on a pending asynchronous query or cancels any pending results. In some cases, it might not be possible to cancel an operation once it is started, and in other cases it might be possible to cancel the operation, but part of its steps might have already been completed.

In situations where you need to create a result set, but do not want to wait until the query engine completes the operation, you can use the **rdAsyncEnable** option with the **OpenResultset** or **Execute** method. This option returns control to your application as soon as the operation is initiated, but before the first row is ready for processing. This gives you an opportunity to execute other code while the query is executed. If you need to stop this operation before it is completed, use the **Cancel** method against the object being created.

The **Cancel** method can also be used against an **rdoConnection** object when you use the **rdAsyncEnable** option to request an asynchronous connection. In this case the attempt to connect to the remote server is abandoned.

You can also use the **Cancel** method against a synchronous **rdoResultset** or **rdoQuery** object to flush remaining result set rows and release resources committed to the query and **rdoResultset**.

If you use the **Cancel** method against **rdoResultset** objects that have multiple result sets pending, *all* result sets are flushed. To simply cancel the *current* set of results and begin processing the next set, use the **MoreResults** method.

Note Using the **Cancel** method against an executing action query might have unpredictable results. If the query is performing an operation that affects a number of rows, some of the rows might be changed, while others are not. For example, if you execute an action query containing an SQL UPDATE statement and use the **Cancel** method before the operation is complete, an indeterminate number of rows are updated — leaving others unchanged. If you intend to use the **Cancel** method against this type of action query, it is recommended that you use transaction methods to rollback or commit partially completed operations.

CancelUpdate Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCancelUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthCancelUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCancelUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCancelUpdateS"}

Cancels any pending updates to an **rdoResultset** object.

Syntax

object.**CancelUpdate**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **CancelUpdate** method flushes the copy buffer and cancels any pending updates from an **Edit** or **AddNew** operation. For example, if a user invokes the **Edit** or **AddNew** method and hasn't yet invoked the **Update** method, **CancelUpdate** cancels any changes made after **Edit** or **AddNew** was invoked. Any information in the copy buffer is lost — that is, any changes made to the row after the **Edit** or **AddNew** methods are invoked, are flushed.

Use the **EditMode** property to determine if there is a pending operation that can be canceled.

If the **CancelUpdate** method is used before using the **Edit** or **AddNew** methods or when the **EditMode** property is set to **rdEditNone**, the method is ignored.

Note Using the **CancelUpdate** method has the same effect as moving to another row without using the **Update** method, except that the current row doesn't change, and various properties, such as **BOF** and **EOF**, aren't updated.

Close Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCloseC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCloseA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCloseS"}

Closes an open remote data object.

Syntax

object.**Close**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Closing an open object removes it from the collection of like objects — except for the **rdoConnection** object. For example, using the **Close** method on an **rdoResultset** removes it from the **rdoResultsets** collection. However, using the **Close** method on the **rdoConnection** object, simply closes and discards any subordinate objects (like **rdoResultset** or **rdoQuery** objects) but does not remove it from the **rdoConnections** collection.

Closing the **rdoConnection** object also releases its parent **rdoEnvironment** object. Any attempt to close the default environment **rdoEnvironments**(0) is ignored. Unlike DAO, RDO collection members cannot be removed with the **Delete** method.

If you try to close an **rdoConnection** object while any **rdoResultset** objects are open, or if you try to close an **rdoEnvironment** object while any **rdoConnection** objects belonging to that specific **rdoEnvironment** are open, those **rdoResultset** objects are closed and any pending updates or edits are rolled back.

If the **rdoConnection** object is defined outside the scope of the procedure, and you exit the procedure without closing it, the **rdoConnection** object remains open until it is explicitly closed or the module in which it is defined is out of scope. Any **rdoResultset** or **rdoQuery** objects that are opened against the **rdoConnection** remain open until explicitly closed. Once all result sets are closed on an **rdoConnection** that is no longer in scope, the **rdoConnection** is closed.

If *object* is already closed when you use **Close**, a trappable error is triggered.

Note Using the **Close** method against an executing action query might have unpredictable results. If the query is performing an operation that affects a number of rows, some of the rows might be changed, while others are not. For example, if you execute an action query containing an SQL UPDATE statement and use the **Close** method before the operation is complete, an indeterminate number of rows are updated — leaving others unchanged. If you intend to use the **Close** method against this type of action query, it is recommended that you use transaction methods to roll back or commit partially completed operations.

ColumnSize Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthColumnSizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthColumnSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthColumnSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthColumnSizeS"}

Returns the number of bytes in an **rdoColumn** object with a data type of **rdTypeLONGVARBINARY** or **rdTypeLONGVARCHAR**.

Syntax

varname = *object* **!** *column***.ColumnSize( )**

The **ColumnSize** method syntax has these parts:

Part	Description
varname	The name of a **Long** or **Variant** variable.
object	An object expression that evaluates to the **rdoResultset** object containing the **rdoColumns** collection.
column	The name of an **rdoColumn** object whose **ChunkRequired** property is set to **True**.

Remarks

Depending on the driver being used, the **ColumnSize** method either returns the size of a binary large object (BLOB) column, or -1 if the size is not available. If the BLOB column size is not available, you can still use the **GetChunk** method to read chunks of data from your BLOB column. The last block has been fetched when the value returned by **GetChunk** is smaller than the size requested (for binary data), at least two bytes smaller than your buffer (for character data), or returns a NULL value.

When working with data types that span multiple database pages, you should use the *chunk* methods to manage the data — but this is not an absolute requirement. You should also use the **GetChunk** and **AppendChunk** methods to manage *chunk* data when the **ChunkRequired** property is **True**. Note that when the size of BLOB data columns is smaller than the **BindThreshold**, it is not necessary to use the chunk methods.

Use the **ColumnSize** method to determine the size of *chunk* columns.

Because the size of a *chunk* data column can exceed 64K, you should assign the value returned by the **GetChunk** method to a variable large enough to store the data returned based on the size returned by the **ColumnSize** method.

Note To determine the size of a non-*chunk* **rdoColumn** object, use the **Size** property.

CreatePreparedStatement Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCreatePreparedStatementC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCreatePreparedStatementA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCreatePreparedStatementS"}

Creates a new **rdoPreparedStatement** object.

Syntax

Set *prepstmt* = *connection*.**CreatePreparedStatement(***name*, *sqlstring***)**

The **CreatePreparedStatement** method syntax has these parts:

Part	Description
prepstmt	An object expression that evaluates to the **rdoPreparedStatement** object you want to create.
connection	An object expression that represents the open **rdoConnection** object.
name	A **String** that is the name of the new **rdoPreparedStatement**. This part is required, but may be an empty string (“”).
sqlstring	A **Variant** expression (a valid SQL statement) that defines the **rdoPreparedStatement**. This part is required, but you can provide an empty string — if you do, you must define the **rdoPreparedStatement** by setting its **SQL** property before executing the new **rdoPreparedStatement**.

Remarks

Note Support for the **rdoPreparedStatement** object is provided in this version of Visual Basic to provide compatibility with previous versions. The **rdoQuery** object should be used as a direct replacement for this object. Because of this, it is also recommended that use of the **CreatePreparedStatement** method be discontinued in favor of the **CreateQuery** method.

Delete Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthDeleteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthDeleteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthDeleteS"}

Deletes the current row in an updatable **rdoResultset** object.

Syntax

object.**Delete**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Delete removes the current row and makes it inaccessible. The deleted row is removed from the **rdoResultset** cursor and the database. When you delete rows from an **rdoResultset**, there must be a current row in the **rdoResultset** before you use **Delete**; otherwise, a trappable error is triggered.

Once you delete a row in an **rdoResultset**, you must reposition the current row pointer to another row in the **rdoResultset** before performing an operation that accesses the current row. Although you can't edit or use the deleted row, it remains current until you reposition to another row. Once you move to another row, however, you can't make the deleted row current again.

When you position to a row in your **rdoResultset** that has been deleted by another user, or if you delete a common row in another **rdoResultset**, a trappable error occurs indicating that the row has been deleted. At this point, the current row is invalid and you must reposition to another valid row. For example, if you use a bookmark to position to a deleted row, a trappable error occurs.

You can undo a row deletion if you use transactions and the **RollbackTrans** method — assuming you use **BeginTrans** before using the **Delete** method.

Using **Delete** produces an error under any of the following conditions:

- There is no current row.
- The connection or **rdoResultset** is read-only.
- No columns in the row are updatable.
- The row has already been deleted.
- Another user has locked the data page containing your row.
- The user does not have permission to perform the operation.

Edit Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthEditC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthEditX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthEditA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthEditS"}

Enables changes to data values in the current row of an updatable **rdoResultset** object.

Syntax

object.**Edit**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Before you use the Edit method, the data columns of an **rdoResultset** are read-only. Executing the **Edit** method copies the current row from an updatable **rdoResultset** object to the copy buffer for subsequent editing. Changes made to the current row's columns are copied to the copy buffer. After you make the desired changes to the row, use the **Update** method to save your changes or the **CancelUpdate** method to discard them. The current row remains current after you use **Edit**.

Caution If you edit a row, and then perform any operation that repositions the current row pointer to another row without first using **Update**, your changes to the edited row are lost without warning. In addition, if you close *object*, or end the procedure which declares the result set or the parent **rdoConnection** object, your edited row might be discarded without warning.

You cannot use the **Edit** method if the **EditMode** property of the **rdoResultset** object indicates that an **Edit** or **AddNew** operation is in progress.

When the **rdoResultset** object's **LockEdits** property setting is **True** (pessimistically locked) , all rows in the **rdoResultset** object's rowset are locked as soon as the cursor is opened and remain locked until the cursor is closed. The number of rows in the rowset is determined by the **RowsetSize** property. Since many remote data sources use page locking schemes, pessimistic locking also locks all data pages of the table(s) containing a row fetched by the **rdoResultset**.

If the **LockEdits** property setting is **False** (optimistically locked), the individual row or the data page containing the row is locked and the new row is compared with the pre-edited row just before it's updated in the database. If the row has changed since you last used the **Edit** method, the **Update** operation fails with a trappable error.

Note Not all data sources use page locking schemes to manage data concurrency. In some cases, data is locked on a row-by-row basis, therefore locks only affect the specific rowset being edited.

Using **Edit** produces an error under any of the following conditions:

- There is no current row.
- The connection or **rdoResultset** is read-only.
- No columns in the row are updatable.
- The **EditMode** property indicates that an **AddNew** or **Edit** is already in progress.
- Another user has locked the row or data page containing your row and the **LockEdits** property is **True**.

Execute Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthExecuteC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthExecuteX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthExecuteA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthExecuteS"}

Runs an action query or executes an SQL statement that does not return rows.

Syntax

connection.**Execute** *source*[, *options*]
query.**Execute** [*options*]

The **Execute** method syntax has these parts:

Part	Description
connection	An object expression that evaluates to the **rdoConnection** object on which the query will run.
query	An object expression that evaluates to the **rdoQuery** object whose **SQL** property setting specifies the SQL statement to execute.
source	A string expression that contains the action query to execute or the name of an **rdoQuery**.
options	A **Variant** or constant that determines how the query is run, as specified in Settings.

Settings

You can use the following constants for the ***options*** argument:

Constant	Value	Description
rdAsyncEnable	32	Execute operation asynchronously.
rdExecDirect	64	Bypass creation of a stored procedure to execute the query. Uses SQLExecDirect instead of SQLPrepare and SQLExecute.

Remarks

It is recommended that you use the **Execute** method only for action queries. Because an action query doesn't return any rows, Execute doesn't return an **rdoResultset**. You can use the **Execute** method on queries that execute multiple statements, but none of these batched statements should return rows. To execute multiple result set queries that are a combination of action and SELECT queries, use the **OpenResultset** method.

Use the **RowsAffected** property of the **rdoConnection** or **rdoQuery** object to determine the number of rows affected by the most recent **Execute** method. **RowsAffected** contains the number of rows deleted, updated, or inserted when executing an action query. When you use the **Execute** method to run an **rdoQuery**, the **RowsAffected** property of the **rdoQuery** object is set to the number of rows affected.

Options

To execute the query asynchronously, use the **rdAsyncEnable** option. If set, the data source query processor immediately begins to process the query and returns to your application before the query is complete. Use the **StillExecuting** property to determine when the query processor is ready to return the results from the query. Use the **Cancel** method to terminate processing of an asynchronous query.

To bypass creation of a temporary stored procedure to execute the query, use the **rdExecDirect** option. This option is required when the query contains references to transactions or temporary tables that only exist in the context of a single operation. For example, if you include a Begin Transaction TSQL statement in your query or reference a temporary table, you must use **rdExecDirect** to ensure that the remote engine is not confused when these objects are left pending at the end of the query.

While it is possible to execute stored procedures using the **Execute** method, it is not recommended because the procedure's return value and output parameters are discarded and the procedure cannot return rows. Use the **OpenResultset** method against an **rdoQuery** to execute stored procedures.

Note When executing stored procedures that do not require parameters, do not include the parenthesis in the SQL statement. For example, to execute the "MySP" procedure use the following syntax: `{Call MySP }`.

Execute Method Example

This example illustrates use of the **Execute** method to execute SQL queries against a remote data source. These action queries do not return rows, but in some cases do return the number of rows affected in the **RowsAffected** property. The example creates a work table called “TestData”, inserts a few rows of data in the table and proceeds to run a DELETE query against the table. Notice that the delete queries have their own embedded transaction management. Because of this, you must use the **rdExecDirect** option to prevent the creation of stored procedures which negate the use of query-provided transactions.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim col As rdoColumn
Dim SQL As String

Private Sub DropRows_Click()
Dim SQL As String, Ans As Integer

SQL = "Begin Transaction Delete TestData " _
   & " Where State = '" & StateWanted & "'"
cn.Execute SQL, rdExecDirect
Ans = MsgBox("Ok to delete these " _
   & cn.RowsAffected & " rows?", vbOKCancel)
If Ans = vbOK Then
   cn.Execute "Commit Transaction", rdExecDirect
Else
   cn.Execute "Rollback Transaction", rdExecDirect
End If
Exit Sub
End Sub

Private Sub Form_Load()
cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=sequel;" _
   & "driver={SQL Server};" _
   & "database=pubs;dsn='';"
cn.EstablishConnection
With qy
   .Name = "TestList"
   .SQL = "Select * from TestData Where State = ?"
   .RowsetSize = 1
   Set .ActiveConnection = cn
End With
SQL = "Drop Table TestData"
cn.Execute SQL

SQL = " CREATE TABLE TestData " _
   & " (ID integer identity NOT NULL, " _
   & " PName char(10) NULL," _
   & " State Char(2) NULL) " _
   & " CREATE UNIQUE INDEX " _
   &  "TestDataIndex on TestData(ID)"
```

```
cn.Execute SQL
SQL = "Insert TestData (PName,State)  " _
   & "Values('Bob', 'CA')" _
   & " Insert TestData (PName,State) " _
   & " Values('Bill', 'WA')" _
   & " Insert TestData (PName,State) " _
   & " Values('Fred', 'WA')” _
   & " Insert TestData (PName,State) " _
   & " Values('George', 'CA')" _
   & " Insert TestData (PName,State) " _
   & " Values('Sam', 'TX')" _
   & " Insert TestData (PName,State) " _
   & " Values('Marilyn', 'TX')"
cn.Execute SQL
Debug.Print cn.RowsAffected
' This returns 1
'(The last INSERT statement affected 1 row)
End Sub

Private Sub SeekRows_Click()
qy(0) = StateWanted
Set rs = qy.OpenResultset(rdOpenForwardOnly, _
rdConcurReadOnly)
List1.Clear
If rs.EOF Then
   MsgBox "No hits for that state"
Exit Sub
End If
Do Until rs.EOF
   List1.AddItem rs!PName & " - " & rs!state
   rs.MoveNext
Loop
End Sub
```

GetChunk Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthGetChunkC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthGetChunkX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthGetChunkA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthGetChunkS"}

Returns all or a portion of the contents of an **rdoColumn** object with a data type of **rdTypeLONGVARBINARY** or **rdTypeLONGVARCHAR**.

Syntax

varname = *object* **!** *column***.GetChunk(***numbytes***)**

The **GetChunk** method syntax has these parts:

Part	Description
varname	The name of a **Variant** that receives the data from the **rdoColumn** object named by *column.*
object	An object expression that evaluates to an **rdoResultset** object containing the **rdoColumns** collection.
column	An object expression that evaluates to an **rdoColumn** object whose **ChunkRequired** property is **True**.
numbytes	A numeric expression that is the number of bytes you want to return.

Remarks

Chunk data columns are designed to store binary or text values that can range in size from a few characters to over 1.2GB and are stored in the database on successive data pages. In most cases, chunk data cannot be managed with a single operation so you must use the chunk methods to save and write data a piece at a time. If the **ChunkRequired** property is **True** for a column, you should use the **GetChunk** and **AppendChunk** methods to manipulate column data. The **BindThreshold** property determines the largest size block that is automatically bound and precludes the need to use the chunk methods.

If the **ChunkRequired** property is **True** for a column, you must use the **GetChunk** method to retrieve the data. The **GetChunk** method moves a portion of the data from a chunk column to a variable. The total number of bytes in the column is determined by executing the **ColumnSize** method.

The **GetChunk** method is used iteratively, copying column data to a variable, one segment or chunk at a time. The chunk size is set by ***numbytes***. The starting point of the copy operation is initially 0, which causes data to be copied from the first byte of the column being read. Subsequent calls to **GetChunk** get data from the first position after the previously read chunk.

The bytes returned by **GetChunk** are assigned to *varname*. Due to memory requirements for the returned data and temporary storage, *numbytes* might be limited, but with 32-bit systems this limitation is over 1.2GB, or more practically the memory and disk capacity of your virtual memory system.

If *numbytes* is greater than the number of bytes in the column, the actual number of bytes in the column is returned. After assigning the results of **GetChunk** to a **Variant** variable, you can use the **Len** function to determine the number of bytes returned.

Use the **AppendChunk** method to write successive blocks of data to the column and **GetChunk** to extract data from the column. Certain operations (copying, for example) involve temporary strings. If string space is limited, you may need to work with smaller segments of a *chunk* column instead of the entire column.

Use the **BindThreshold** property to specify the largest column size that will be automatically bound.

Note Because the size of a chunk data column can exceed 1.2GB, you should assign the value

returned by the **GetChunk** method to a variable large enough to store the data returned based on the size returned by the **ColumnSize** method.

GetRows Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthGetrowsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthGetrowsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthGetrowsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthGetrowsS"}

Retrieves multiple rows of an **rdoResultset** into an array.

Syntax

array = *object*.**GetRows (***rows***)**

The **GetRows** method syntax has these parts:

Part	Description
array	The name of a **Variant** type variable to store the returned data.
object	An object expression that evaluates to an object in the Applies To list.
rows	A **Long** value indicating the number of rows to retrieve.

Remarks

Use the **GetRows** method to copy one or more entire rows from an **rdoResultset** into a two-dimensional array. The first array subscript identifies the column and the second identifies the row number, as follows:

```
avarRows(intColumn)(intRow)
```

To get the first column value in the second row returned, use the following:

```
col1 = avarRows(0,1)
```

To get the second column value in the first row, use the following:

```
col2 = avarRows(1,0)
```

If more rows are requested than are available, only the available rows are returned. Use **Ubound** to determine how many rows are actually fetched, as the array is resized based on the number of rows returned. For example, if you return the results into a **Variant** called varA, you could determine how many rows were actually returned by using:

```
numReturned = Ubound(varA,2) + 1
```

The "+ 1" is used because the first data returned is in the 0th element of the array. The number of rows that can be fetched is constrained by available memory and should be chosen to suit your application — don't expect to use **GetRows** to bring your entire table or result set into an array if it is a large table.

GetRows does not return data from columns whose **ChunkRequired** property is **True** — a variant value containing an ODBC S-code is returned in these columns instead.

After a call to **GetRows**, the current row is positioned at the next unread row. That is, **GetRows** is equivalent to using the **Move** (*rows*) method.

If you are trying to fetch all the rows using multiple **GetRows** calls, use the **EOF** property to determine if there are rows available. **GetRows** returns less than the number requested either at the end of the **rdoResultset**, or if it cannot fetch a row in the range requested. For example, if a fifth row cannot be retrieved in a group of ten rows that you're trying to fetch, **GetRows** returns four rows and leaves currency on the row that caused the problem. It will not generate a run-time error.

The **GetRows** method fetches data from the ODBC buffers based on the **RowsetSize** property. RDO proceeds to fetch from the current row toward the end of the result set — returning as many rows as you requested. As the current rowset is exhausted, RDO issues another **SQLExtendedFetch** function call to fetch subsequent rowsets from the database. This technique applies to all types of cursors.

GetRows Method Example

This example illustrates use of the **GetRows** method to fetch rows from an **rdoResultset** into a variant array. The code opens a connection to a remote data source and creates an **rdoQuery** object that requires a single parameter. The `GetRowsNow` procedure executes the query with a user-supplied parameter and uses **GetRows** to fetch the rows from the result set.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim RowBuf As Variant
Dim RowsReturned As Integer
Dim i As Integer
Dim Ans As Integer

Private Sub GetRowsNow_Click()
qy(0) = StateWanted
rs.Requery

Do Until rs.EOF
   List1.Clear
   RowBuf = rs.GetRows(5)      'Get the next 5 rows
   RowsReturned = UBound(RowBuf, 2) + 1
   For i = 0 To RowsReturned - 1
      List1.AddItem RowBuf(0, i) & ":" & RowBuf(1, i)
   Next i
   Ans = MsgBox("Press Ok to see next 5 rows " _
      &" or Cancel to quit", vbOKCancel)
   If Ans = vbOK Then Else Exit Sub
Loop
End Sub

Private Sub Form_Load()
cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=SEQUEL;" _
   driver={SQL Server};database=pubs;dsn='';"
cn.EstablishConnection
With qy
   .Name = "GetRowsQuery"
   .SQL = "Select * from Titles T, Publishers P " _
   & " Where T.Pub_ID = P.Pub_ID " _
   & " and P.State = ?"
   .RowsetSize = 1
   Set .ActiveConnection = cn
   .rdoParameters(0) = "CA"
   Set rs = .OpenResultset(rdOpenKeyset, _
      rdConcurRowver)
End With
End Sub
```

MoreResults Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthMoreResultsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthMoreResultsX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthMoreResultsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthMoreResultsS"}

Clears the current result set of any pending rows and returns a **Boolean** value that indicates if one or more additional result sets are pending.

Syntax

variable = *object*.**MoreResults**

The **MoreResults** method syntax has these parts:

Part	Description
variable	A Boolean variable that indicates if additional result sets are found as described in Return Values.
object	An object expression that evaluates to an open **rdoResultset** object variable.

Return Values

The return values for *variable* are:

Value	Description
True	Additional result sets are ready to be processed.
False	All result sets in the **rdoResultset** have been processed.

Remarks

Calling this method will flush the current result set, call the ODBC API **SQLMoreResults** function to see if there is another result set on the same statement, and if there is, loads the new result set, positions the current row pointer at the first row and returns **True**. If there are no more result sets, this method will return **False**, and both the **EOF** and **BOF** properties will be **True**.

If the result set was created asynchronously (developer used **rdAsyncEnable** in the ***Options*** parameter), the **MoreResults** method will be executed asynchronously as well. You should use the **StillExecuting** property to determine when the next result set has been enabled. Asynchronous execution of the **MoreResults** method follows the same rules as asynchronously opening a result set.

When the query used to create an **rdoResultset** returns more than one result set, use the **MoreResults** method to end processing of the current result set and test for subsequent result sets. If there are no additional result sets to process, the **MoreResults** method returns **False** and both **BOF** and **EOF** are set to **True**. In any case, using the **MoreResults** method flushes the current **rdoResultset**.

You can also use the **Cancel** method to flush the contents of an **rdoResultset**. However, **Cancel** also flushes any additional result sets not yet processed.

Not all cursor libraries support multiple resultset queries. For example, the Server-side cursor library does not support this type of query unless you disable the cursor processor by requesting a forward-only, read-only cursor with a **RowsetSize** property of 1.

MoreResults Method Example

The following example illustrates use of the **MoreResults** method. In this example, an SQL query containing three separate SELECT queries is executed. The first query simply returns the number of publishers in the *Publishers* table. The next two queries each return two columns resulting from more complex join operations. All of this information is displayed in a **ListBox** control.

```
Option Explicit
Dim Cn As New rdoConnection
Dim Rs As rdoResultset
Dim SQL As String

Private Sub Test_Click()
SQL = "Select Count(*) From Publishers" _
   & " Select Pub_Name, Title  " _
   & " From Publishers P, Titles T" _
   & " Where P.Pub_ID = T.Pub_ID" _
   & " Select Au_Lname, Title " _
   & " From Titles T, TitleAuthor Ta, Authors A" _
   & " Where T.title_ID = ta.Title_ID " _
   & " and Ta.Au_ID = A.Au_ID"
Set Rs = Cn.OpenResultset(SQL, rdOpenForwardOnly, _
   rdConcurReadOnly)
' From the first set of results
List1.AddItem "Publishers: " & Rs(0)
'
'   Loop through all of the remaining result sets
'
Do While Rs.MoreResults
   List1.AddItem Rs(0).Name & " - " & Rs(1).Name
   Do Until Rs.EOF
      List1.AddItem Rs(0) & "     -    " & Rs(1)
      Rs.MoveNext
   Loop
Loop
End Sub

Private Sub Form_Load()
On Error GoTo CnEh
With Cn
   .Connect = "UID=;PWD=;Database=Pubs;" _
   & "Server=SEQUEL;Driver={SQL Server}" _
   & "DSN='';"
   .LoginTimeout = 5
   .CursorDriver = rdUseOdbc
   .EstablishConnection rdDriverNoPrompt, True
End With
Exit Sub

CnEh:
Dim er As rdoError
Debug.Print Err, Error
For Each er In rdoErrors
   Debug.Print er.Description, er.Number
Next er
Resume Next
End Sub
```

Move Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthMoveC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthMoveA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthMoveS"}

Repositions the current row pointer in an **rdoResultset** object.

Syntax

object.**Move** *rows*[, *start*]

The **Move** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
rows	A signed **Long** value that specifies the number of rows the position will move as described in Settings.
start	A **Variant** value that identifies a bookmark as described in Settings.

Settings

If ***rows*** is greater than 0, the position is moved forward (toward the end of the cursor). If ***rows*** is less than 0, the position is moved backward (toward the beginning of the cursor). If ***rows*** is equal to 0, any pending edits are discarded and the current row is refreshed from the data source. At a lower level, when you use 0 as the ***rows*** argument, RDO executes the ODBC **SQLExtendedFetch** function to re-fetch the current rowset (as determined by the **RowsetSize** property) from the database.

If ***start*** is specified, the move begins relative to this bookmark. If ***start*** is not specified, **Move** begins from the current row.

Remarks

If using **Move** repositions the current row to a position before the first row, the position is moved to the beginning-of-file (**BOF**) position. If the **rdoResultset** contains no rows and its **BOF** property is set to **True**, using this method to move backward triggers a trappable run-time error. If either the **BOF** or **EOF** property is **True** and you attempt to use the **Move** method without a valid bookmark, a trappable error is triggered.

If using **Move** repositions the current row to a position after the last row, the position is moved to the end-of-file (**EOF**) position. If the **rdoResultset** contains no rows and its **EOF** property is set to **True**, then using this method to move forward produces a trappable run-time error.

If you use **Move** on an **rdoResultset** object based on an SQL-specific query or **rdoQuery**, the query is forced to completion and the **rdoResultset** object is fully populated.

If you use any method that repositions the current row pointer after using the **Edit** or **AddNew** method but before using the **Update** method, any changes made to the copy buffer are lost.

To make the first, last, next, or previous row in an **rdoResultset** the current row, use the **MoveFirst**, **MoveLast**, **MoveNext**, or **MovePrevious** method. To position the current row pointer based on an absolute row number, use the **AbsolutePosition** property. To position the current row pointer based on a percentage of the accessed rows of a result set, use the **PercentPosition** property.

When you use the **Move** method or any other method to reposition the current row pointer, the RowCurrencyChange event is fired.

When using a forward-only **rdoResultset**, the you can reposition the current row *only* by using the **MoveNext** method. You cannot use the **MoveLast**, **MovePrevious**, **MoveFirst**, or **Move** method, or

the **PercentPosition** or **AbsolutePosition** property, to reposition the current row pointer.

MoveFirst, MoveLast, MoveNext, MovePrevious Methods (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthMoveFirstC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthMoveFirstX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthMoveFirstA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthMoveFirstS"}

Repositions the current row pointer to the first, last, next, or previous row in a specified **rdoResultset** object and makes that row the current row.

The syntax for these methods have these parts:

object.{**MoveFirst** | **MoveNext** | **MovePrevious**}
*object***. MoveLast** ([*Options* as Variant])

Part	Description
object	An object expression that evaluates to a **rdoResultset** object.
options	A **Variant** or constant that determines how the operation is carried out, as specified in Settings.

Settings

You can use the following constant for the ***options*** argument:

Constant	Value	Description
rdAsyncEnable	32	Execute operation asynchronously.

Remarks

Use the *Move* methods to reposition the current row pointer from row to row without applying a condition.

If you specify the **rdAsyncEnable** option with the **MoveLast** method, the move operation is executed asynchronously. That is, control is returned to your application immediately — often before the operation has completed. This prevents your application from blocking until the operation is complete. To check for completion of the operation, you can either wait for the QueryComplete or RowCurrencyChange events, or periodically check the **StillExecuting** property which returns **False** when the move operation is complete.

Caution If you edit the current row, be sure to save the changes using the **Update** method before you move to another row. If you move to another row without updating, your changes are lost without warning.

When you open the result set named by *object*, the first row is current and the **BOF** property is set to **False**. If the result set contains no rows, the **BOF** property is set to **True**, and there is no current row.

If the first or last row is already current when you use **MoveFirst** or **MoveLast**, the current row doesn't change.

If you use **MovePrevious** when the first row is current, the **BOF** property is set to **True**, and there is no current row. If you use **MovePrevious** again, an error occurs; **BOF** remains **True**.

If you use **MoveNext** when the last row is current, the **EOF** property is set to **True**, and there is no current row. If you use **MoveNext** again, an error occurs; **EOF** remains **True**.

If you use **MoveLast** on an **rdoResultset** object based on an SQL-specific query or **rdoQuery**, the query is forced to completion and the **rdoResultset** object is fully populated.

If you use any method that repositions the current row pointer after using the **Edit** or **AddNew** method but before using the **Update** method, any changes made to the copy buffer are lost.

To move the position of the current row in an **rdoResultset** object a specific number of rows forward

or backward, use the **Move** method.

To position the current row pointer based on an absolute row number, use the **AbsolutePosition** property. To position the current row pointer based on a percentage of the accessed rows of a result set, use the **PercentPosition** property.

When you use the **Move** method or any other method to reposition the current row pointer, the RowCurrencyChange event is fired.

When using a forward-only **rdoResultset**, the you can reposition the current row *only* by using the **MoveNext** method. You cannot use the **MoveLast**, **MovePrevious**, **MoveFirst**, or **Move** method, or the **PercentPosition** or **AbsolutePosition** property, to reposition the current row pointer. If you use one of the prohibited Move methods on a forward-only result set, your code will trip an ODBC "Fetch type out of range" error.

Move Methods Example

This example illustrates use of the **rdAsyncEnable** option in conjunction with the **MoveLast** method. The *Phones* table is simply a table with over 15,000 rows which takes some time to process. While this is not a recommended technique, it provides a way to illustrate a query that takes a significant length of time to run and fully populate — as is done when you execute the **MoveLast** method. The application uses a status bar to indicate the degree of completion of the operations.

```
Option Explicit
Dim rdoCn As New rdoConnection
Dim rdoRs As rdoResultset
Dim SQL As String
Dim TimeExpected As Single
Dim Ts As Single, Tn As Single

Private Sub Command1_Click()
TimeExpected = 5      ' We expect this to take about 5 seconds
SQL = "Select Email, Name From Phones"
Set rdoRs = rdoCn.OpenResultset(Name:=SQL, _
   Type:=rdOpenStatic, _
   LockType:=rdConcurReadOnly, _
   Option:=rdAsyncEnable)
ShowProgress "Query"
    '
'   Query Has completed... now move to the last row
    '
rdoRs.MoveLast rdAsyncEnable
' We expect this to take about 15 seconds
TimeExpected = 15
ShowProgress "MoveLast"
rdoCn.Close
End Sub

Sub ShowProgress(Operation As String)
Ts = Timer
'  time to execute  query
ProgressBar1.Max = TimeExpected
While rdoRs.StillExecuting
   Tn = Int(Timer - Ts)
   If Tn < TimeExpected Then
      ProgressBar1 = Tn
   Else
      ProgressBar1.Max = ProgressBar1.Max + 10
      TimeExpected = ProgressBar1.Max
   End If
   DoEvents
Wend
Status = Operation & "Done. Duration:"  _
   & Int(Timer - Ts)
End Sub
Private Sub Form_Load()
With rdoCn
   .Connect = "UID=;PWD=;Database=WorkDB;" _
   & "Server=BETAV486;Driver={SQL Server}" _
   & "DSN='';"
   .LoginTimeout = 5
```

```
   .EstablishConnection rdDriverNoPrompt, True
End With
Exit Sub
End Sub
```

OpenConnection Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthOpenConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthOpenConnectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthOpenConnectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthOpenConnectionS"}

Opens a connection to an ODBC data source and returns a reference to the **rdoConnection** object that represents a specific database.

Syntax

Set *connection* = *environment*.**OpenConnection(***dsName*[, *prompt*[, *readonly*[, *connect*[, *options*]]]]**)**

The **OpenConnection** method syntax has these parts:

Part	Description
connection	An object expression that evaluates to an **rdoConnection** object that you're opening.
environment	An object expression that evaluates to an existing **rdoEnvironment** object. You must provide an **rdoEnvironment** object.
dsName	A string expression that is the name of a registered ODBC data source name or a zero-length string ("") as described in Settings.
prompt	A **Variant** or constant that determines how the operation is carried out, as specified in Settings.
readonly	A **Boolean** value that is **True** if the connection is to be opened for read-only access, and **False** if the connection is to be opened for read/write access. If you omit this argument, the connection is opened for read/write access.
connect	A string expression used to pass arguments to the ODBC driver manager for opening the database — the *connect string* as described in Settings.
options	A **Variant** or constant that determines how the operation is carried out, as specified in Settings.

Settings

The ***connect*** argument constitutes the ODBC connect arguments, and is dependent on the ODBC driver. See the **Connect** property for syntax and typical settings. If the ***connect*** argument is an empty string (""), the user name and password are taken from the **rdoEnvironment** object's **UserName** and **Password** properties, and a ***dsName*** argument *must* be provided.

If the provided ***dsName*** doesn't refer to a valid ODBC data source name, and the Data Source Name (DSN) parameter does not appear in the ***connect*** argument an error occurs if *prompt* is **rdDriverNoPrompt**; otherwise, the user is prompted to select from a list of registered data source names. If ***dsName*** is a zero-length string, the connect string must indicate the driver and server names.

Based on the ***prompt*** value, the ODBC driver manager exposes a dialog which prompts the user for connection information such as DSN, user name, and password. Use one of the following constants that defines how the user should be prompted:

***prompt* Constant**	**Value**	**Description**
rdDriverPrompt	0	The driver manager displays the ODBC Data Sources dialog box. The connection string used to establish the connection is

		constructed from the DSN selected and completed by the user via the dialog boxes, or, if no DSN is chosen and the **DataSourceName** property is empty (in the case of the **RemoteData control**), the default DSN is used.
rdDriverNoPrompt	1	The driver manager uses the connection string provided in *dsName* and *connect*. If sufficient information is not provided, the **OpenConnection** method returns a trappable error.
rdDriverComplete	2	(Default) If the connection string provided includes the DSN keyword, the driver manager uses the string as provided in *connect*. Otherwise it behaves as it does when **rdDriverPrompt** is specified.
rdDriverComplete Required	3	Behaves like **rdDriverComplete** except the driver disables the controls for any information not required to complete the connection. If the controls are disabled, users cannot select or specify missing arguments.

You can use the following constant for the ***options*** argument:

***options* Constant**	**Value**	**Description**
rdAsyncEnable	32	Execute operation asynchronously.

Remarks

When you successfully execute the **OpenConnection** method, a new **rdoConnection** object is instantiated and added to the **rdoConnections** collection, and a network connection is established to the remote server. If the connection cannot be established, no object is created and a trappable error is fired.

Note RDO 2.0 behaves differently than RDO 1.0 in how it handles orphaned references to **rdoConnection** objects. When you Set a variable already assigned to an **rdoConnection** object with another **rdoConnection** object using the **OpenConnection** method, the existing **rdoConnection** object is closed and dropped from the **rdoConnections** collection. In RDO 1.0, the existing object remained open and was left in the **rdoConnections** collection.

If you set the ***options*** argument to **rdAsyncEnable**, the connection operation is executed asynchronously. That is, control returns to your application before the connection has been established to prevent your application from blocking while the connection is being made. You can check for completion of the connection by polling the **rdoConnection** object's **StillConnecting** property, which returns **False** when the connection operation is complete. You can also code an event procedure for the Connect event which is fired when the connect operation is complete. If you use the **Cancel** method while waiting for an asynchronous connection to be established, the connection attempt is abandoned.

Before the process of establishing a connection is started, the BeforeConnect event is fired. This event procedure permits you to examine and modify the connect string and prompt levels as needed.

There are a variety of reasons why a connection might not be made. These include but are not limited to the following:

- Lack of proper user ID and password.
- Incorrect driver or options configuration.

- Lack of correct network or server permissions.
- The remote server could not be found on the network, or is not operating.
- The remote server did not have sufficient resources or connections to permit another user to connect.

DSN-Less Connections

In some cases, it might not be necessary to create and register a Data Source Name (DSN) before attempting to open a connection to a data source. If your remote server uses the named pipes LAN protocol and the default OEMTOANSI settings, you can simply provide the name of the server and ODBC driver in the connect string. You must provide an empty DSN entry in the connect string, or in the dsName parameter as the ***last*** argument. If the ODBC driver manager finds a null DSN entry, it attempts to locate it unless it has already determined the driver and server values. The connect string shown below is used to establish a DSN-less connection to a SQL Server named “BETAV486”:

```
Connect = "UID=;PWD=;Database=WorkDB;" _
& "Server=BETAV486;Driver={SQL Server}" _
& "DSN='';"
```

Other Connect String Options

Establishing an **rdoConnection** may require that the user specified by the **UserName** property, or UID connect string argument have permission to access the network, the specific data source server, and the chosen database on that server. Failure to meet these qualifications might result in failure to connect.

If you do not specify a database either through the DATABASE parameter of the ***connect*** argument or through the data source entry, the database opened when you establish a connection is determined by the default database assigned to the user by the database administrator. In some cases, you can change the default database by executing an action query containing an SQL command such as the Transact SQL USE *database* statement.

Note The *connect* part of the **OpenConnection** method is coded differently than the *source* part of the **OpenDatabase** method as used with DAO. The *connect* part neither requires nor supports use of the “ODBC;” keyword at the beginning of the connect string. In addition, the *connect* part does not support use of the LOGINTIMEOUT argument – use the **LoginTimeout** property of the **rdoEnvironment** object instead.

Use the **Close** method on the object to close a database associated with an **rdoConnection**, remove the connection from the **rdoConnections** collection, and disconnect from the data source.

You can also declare a new **rdoConnection** object using the **Dim** statement as follows:

```
Dim myCn as New rdoConnection
```

Once instantiated in this manner, you can set the **rdoConnection** properties as required, and use the **EstablishConnection** method to open the connection.

For more information about ODBC drivers and the specific connect string arguments they require, see the Help file provided with the driver.

OpenResultset Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthOpenResultsetC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthOpenResultsetX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthOpenResultsetA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthOpenResultsetS"}

Creates a new **rdoResultset** object.

Syntax

Set *variable* = *connection*.**OpenResultset**(*name* [,*type* [,*locktype* [,*option*]]])
Set *variable* = *object*.**OpenResultset**([*type* [,*locktype* [, *option*]]])

The **OpenResultset** method syntax has these parts:

Part	Description
variable	An object expression that evaluates to an **rdoResultset** object.
connection	An object expression that evaluates to an existing **rdoConnection** object you want to use to create the new **rdoResultset**.
object	An object expression that evaluates to an existing **rdoQuery** or **rdoTable** object you want to use to create the new **rdoResultset**.
name	A **String** that specifies the source of the rows for the new **rdoResultset**. This argument can specify the name of an **rdoTable** object, the name of an **rdoQuery**, or an SQL statement that might return rows.
type	A **Variant** or constant that specifies the type of cursor to create as indicated in Settings.
locktype	A **Variant** or constant that specifies the type of concurrency control. If you don't specify a *locktype*, **rdConcurReadOnly** is assumed.
option	A **Variant** or constant that specifies characteristics of the new **rdoResultset**.

Settings

- ***name***

  The ***name*** argument is used when the **OpenResultset** method is used against the **rdoConnection** object, and no query has been pre-defined. In this case, name typically contains a row-returning SQL query. The query can contain more than one SELECT statement, or a combination of action queries and SELECT statements, but not just action queries, or a trappable error will result. See the **SQL** property for additional details.

- Cursor ***type***

  Note Not all types of cursors and concurrency are supported by every ODBC data source driver. See **rdoResultset** for more information. In addition, not all types of cursor drivers support SQL statements that return more than one set of results. For example, server-side cursors do not support queries that contain more than one SELECT statement.

  The ***type*** argument specifies the type of cursor used to manage the result set. If you don't specify a type, **OpenResultset** creates a forward-only **rdoResultset**. Not all ODBC data sources or drivers can implement all of the cursor types. If your driver cannot implement the type chosen, a warning message is generated and placed in the **rdoErrors** collection. Use one of the following result set type constants that defines the cursor type of the new **rdoResultset** object. For

additional details on types of cursors, see the **CursorType** property.

***type* Constant**	**Value**	**Description**
rdOpenForwardOnly	0	(Default) Opens a forward-only-type **rdoResultset** object.
rdOpenKeyset	1	Opens a keyset-type **rdoResultset** object.
rdOpenDynamic	2	Opens a dynamic-type **rdoResultset** object.
rdOpenStatic	3	Opens a static-type **rdoResultset** object.

- Concurrency *LockType*

  In order to maintain adequate control over the data being updated, RDO provides a number of concurrency options that control how other users are granted, or refused access to the data being updated. In many cases, when you lock a particular row using one of the **LockType** settings, the remote engine might also lock the entire page containing the row. If too many pages are locked, the remote engine might also escalate the page lock to a table lock to improve overall system performance.

  Not all lock types are supported on all data sources. For example, for SQL Server and Oracle servers, static-type **rdoResultset** objects can only support **rdConcurValues** or **rdConcurReadOnly**. For additional details on the types of concurrency, see the **LockType** property.

***locktype* Constant**	**Value**	**Description**
rdConcurReadOnly	1	(Default) Read-only .
rdConcurLock	2	Pessimistic concurrency.
rdConcurRowVer	3	Optimistic concurrency based on row ID.
rdConcurValues	4	Optimistic concurrency based on row values.
rdConcurBatch	5	Optimistic concurrency using batch mode updates. **Status** values returned for each row successfully updated.

- Other *options*

  If you use the **rdAsyncEnable** option, control returns to your application as soon as the query is begun, but before a result set is available. To test for completion of the query, use the **StillExecuting** property. The **rdoResultset** object is not valid until **StillExecuting** returns **False**. You can also use the QueryComplete event to determine when the query is ready to process. Until the **StillExecuting** property returns **True**, you cannot reference any other property of the uninitialized **rdoResultset** object and only the **Cancel** and **Close** methods are valid.

  If you use the **rdExecDirect** option, RDO uses the *SQLExecDirect* ODBC API function to execute the query. In this case, no temporary stored procedure is created to execute the query. This option can save time if you don't expect to execute the query more than a few times in the course of your application. In addition, when working with queries that should not be run as stored procedures but executed directly, this option is mandatory. For example, in queries that create temporary tables for use by subsequent queries, you must use the **rdExecDirect** option.

  You can use the following constants for the ***options*** argument:

Constant	**Value**	**Description**
rdAsyncEnable	32	Execute operation asynchronously.
rdExecDirect	64	Bypass creation of a stored procedure to execute the query. Uses SQLExecDirect instead of

SQLPrepare and SQLExecute.

Remarks

If the **OpenResultset** method succeeds, RDO instantiates a new **rdoResultset** object and appends it to the **rdoResultsets** collection – even if no rows are returned by the query. If the query fails to compile or execute due to a syntax error, permissions problem or other error, the **rdoResultset** is not created and a trappable error is fired. The **rdoResultset** topic contains additional details on **rdoResultset** behavior and managing the **rdoResultsets** collection.

Note RDO 2.0 behaves differently than RDO 1.0 in how it handles orphaned references to **rdoResultset** objects. When you Set a variable already assigned to an **rdoResultset** object with another **rdoResultset** object using the **OpenResultset** method, the existing **rdoResultset** object is closed and dropped from the **rdoResultsets** collection. In RDO 1.0, the existing object remained open and was left in the **rdoResultsets** collection.

Note Before you can use the name of a base table in the ***name*** argument, you must first use the **Refresh** method against the **rdoTables** collection to populate it. You can also populate the **rdoTables** collection by referencing one of its members by its ordinal number. For example, referencing **rdoTables**(0) will populate the entire collection.

Executing Multiple Operations on a Connection

If there is an unpopulated **rdoResultset** pending on a data source that can only support a single operation on an **rdoConnection** object, you cannot create additional **rdoQuery** or **rdoResultset** objects using the **OpenResultset** method**,** or use the **Refresh** method on the **rdoTable** object until the **rdoResultset** is flushed, closed, or fully populated. For example, when using SQL Server 4.2 as a data source, you cannot create an additional **rdoResultset** object until you move to the last row of the last result set of the current **rdoResultset** object. To populate the result set, use the **MoreResults** method to move through all pending result sets, or use the **Cancel** or **Close** method on the **rdoResultset** to flush all pending result sets.

rdoCreateEnvironment Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthrdoCreateEnvironmentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthrdoCreateEnvironmentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthrdoCreateEnvironmentS"}

Creates a new **rdoEnvironment** object.

Syntax

Set *variable* = **rdoCreateEnvironment(***name*, *user*, *password***)**

The **rdoCreateEnvironment** method syntax has these parts:

Part	Description
variable	An object expression that evaluates to an **rdoEnvironment** object.
name	A **String** variable that uniquely names the new **rdoEnvironment** object. See the **Name** property for details on valid **rdoEnvironment** names.
user	A **String** variable that identifies the owner of the new **rdoEnvironment** object. See the **UserName** property for more information.
password	A **String** variable that contains the password for the new **rdoEnvironment** object. The password can be up to 14 characters long and can include any characters except ASCII character 0 (null).

Remarks

The **rdoEnvironment** object defines a transaction, user, and password context. When the **rdoEngine** is initialized, a default **rdoEnvironments**(0) is created automatically with the **rdoDefaultUser** and **rdoDefaultPassword** user name and password. If you need to define alternate transaction scopes that contain specific **rdoConnection** objects, or specific users, use the **rdoCreateEnvironment** method and specify the specific users for the environment. You can then open connections against this new environment.

Unlike the other methods you use to create Remote Data Objects, **rdoCreateEnvironment** requires that you provide all of its parts. If you don't provide all of the parts, the object won't be added to the collection. In addition, **rdoEnvironment** objects aren't permanent and can't be saved. Once you create an **rdoEnvironment** object, you can only modify the **UserName** and **Timeout** property settings.

You don't have to append the new **rdoEnvironment** object to a collection before you can use it — it is automatically appended to the **rdoEnvironments** collection.

If *name* refers to an object that is already a member of the **rdoEnvironments** collection, a trappable error occurs.

Once you use **rdoCreateEnvironment** to create a new **rdoEnvironment** object, an **rdoEnvironment** session is started, and you can refer to the **rdoEnvironment** object in your application.

To remove an **rdoEnvironment** object from the **rdoEnvironments** collection, use the **Close** method on the **rdoEnvironment** object. You cannot remove **rdoEnvironments**(0).

rdoRegisterDataSource Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthrdoRegisterDataSourceC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthrdoRegisterDataSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthrdoRegisterDataSourceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthrdoRegisterDataSourceS"}

Enters connection information for an ODBC data source into the Windows Registry.

Syntax

rdoRegisterDataSource *DSN*, *driver*, *silent*, *attributes*

The **rdoRegisterDataSource** method syntax has these parts:

Part	Description
DSN	A string expression that is the name used in the **OpenConnection** method that refers to a block of descriptive information about the data source. For example, if the data source is an ODBC remote database, it could be the name of the server.
driver	A string expression that is the name of the ODBC driver. This isn't the name of the ODBC driver dynamic link library (DLL) file. For example, *SQL Server* is a driver name, but *SQLSRVR.DLL* is the name of a DLL file. You must have ODBC and the appropriate driver already installed.
silent	A **Boolean** value that is **True** if you don't want to display the ODBC driver dialog boxes that prompt for driver-specific information, or **False** if you do want to display the ODBC driver dialog boxes. If *silent* is **True**, *attributes* must contain all the necessary driver-specific information or the dialog boxes are displayed anyway.
attributes	A string expression that is a list of keywords to be added to the ODBC.INI file. The keywords are in a carriage-return-delimited string.

Remarks

When you use the **OpenConnection** or **EstablishConnection** method, you can use a registered data source entry to provide connection information.

If the data source is already registered in the Windows Registry when you use the **rdoRegisterDataSource** method, the connection information is updated. If the **rdoRegisterDataSource** method fails for any reason, no changes are made to the Windows Registry, and an error occurs.

For more information about ODBC drivers such as SQL Server, see the Help file provided with the driver.

Note You are encouraged to use the Windows Control Panel 32-bit ODBC Data Sources dialog box to add new data sources, or to make changes to existing entries.

Microsoft SQL Server Attributes

The following attributes are used when setting up DSN entries for Microsoft SQL Server drivers as extracted from the Drvssrvr.Hlp file. Other vendor's drivers expose their own set of attributes that might or might not conform to this set. See the documentation provided with your driver for additional details.

Keyword	Description
ADDRESS	The network address of the SQL Server database management system from which the driver retrieves data.
DATABASE	The name of the SQL Server database.
DESCRIPTION	A description of the data in the data source.
LANGUAGE	The national language to be used by SQL Server.
NETWORK	The network library connecting the platforms on which SQL Server and the SQL Server driver reside.
OEMTOANSI	Enables conversion of the OEM character set to the ANSI character set if the SQL Server client machine and SQL Server are using the same non-ANSI character set. Valid values are YES for on (conversion is enabled) and NO for off.
SERVER	The name of the network computer on which the data source resides.
TRANSLATIONDLL	The name of the DLL that translates data passing between an application and a data source.
TRANSLATIONNAME	The name of the translator that translates data passing between an application and a data source.
TRANSLATIONOPTION	Enables translation of data passing between an application and a data source.
USEPROCFORPREPARE	Disables generation of stored procedures for SQLPrepare. Valid values are NO for off (generation is disabled) and YES for on. The default value (set in the setup dialog box) is YES.

Setting the OEMTOANSI Option

If the SQL Server client computer and SQL Server are using the same non-ANSI character set, select this option. For example, if SQL Server uses code page 850 and this client computer uses code page 850 for the OEM code page, selecting this option will ensure that extended characters stored in the database are property converted to ANSI for use by Windows-based applications.

When this option is set to YES and the SQL Server client machine and SQL Server are using different character sets, you must specify a character set translator.

Setting the Server Option

The Server option sets the name of the server. “(local)” can be entered as the server on a Microsoft Windows NT computer if the DSN is intended to reference a server on the local system. The user can then use a local copy of SQL Server (that listens on named pipes), even when running a non-networked version of SQL Server. Note that when the 16-bit SQL Server driver is using “(local)” without a network, the MS Loopback Adapter must be installed.

For more information about server names for different types of networks, see Microsoft SQL Server Setup.

Setting the Address Option

The Address option sets the network pathname address of the SQL Server database management system (DBMS) from which the driver retrieves data. For Microsoft SQL Server you can usually omit this argument when sets it to (Default).

Setting the Network Option

The Network attribute sets the name of the 32-bit SQL Server Net-Library DLL that the SQL Server driver uses to communicate with the network software. If this option is not provided, the SQL Server driver uses the client computer's default Net-Library, which is specified in the Default Network box in the Net-Library tab of the SQL Server Client Configuration Utility.

If you create a data source using a Network Library and optionally a Network Address, ODBC SQL Server Setup will create a server name entry that you can see in the Advanced tab in the SQL Server Client Configuration Utility. These server name entries can also be used by DB-Library applications.

rdoRegisterDataSource Method Example

The following example illustrates use of the **rdoRegisterDataSource** method to create a new ODBC data source entry.

```
Private Sub RegisterDataSource()
Dim en As rdoEnvironment
Dim cnTest As rdoConnection
Dim strAttribs As String
' Build keywords string.
strAttribs = "Description=" _
        & "SQL Server on server SEQUEL" _
  & Chr$(13) & "OemToAnsi=No" _
  & Chr$(13) & "SERVER=SEQUEL" _
  & Chr$(13) & "Network=DBNMPNTW" _
  & Chr$(13) & "Database=WorkDB" _
  & Chr$(13) & "Address=\\SEQUEL\PIPE\SQL\QUERY"

' Create new registered DSN.
rdoEngine.rdoRegisterDataSource "Example", _
         "SQL Server", True, strAttribs
' Open the database.
Set en = rdoEngine.rdoEnvironments(0)
Set cnTest = en.OpenConnection( _
  dsname:="Example", _
  Prompt:=rdDriverNoPrompt, _
  Connect:="UID=;PWD=;")


End Sub
```

Refresh Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthRefreshC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthRefreshX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthRefreshA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthRefreshS"}

Closes and rebuilds the **rdoResultset** object created by a **RemoteData control** or refreshes the members of the collections in the Applies To list.

Syntax

object.**Refresh**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

You can use the **Refresh** method on a **RemoteData** control to close and reopen the **rdoResultset** if the properties that describe the result set have changed. When you use the **Refresh** method, the properties and current row position is reset to the state set when the query was first run.

Once the **Refresh** method has been executed against the **RemoteData** control, all stored **rdoResultset** bookmarks are invalid.

If both the **BOF** and **EOF** property settings of the **rdoResultset** object are **True**, or the **RowCount** property is set to 0 after you use the **Refresh** method, the query didn't return any rows and the new **rdoResultset** contains no data.

Use the **Refresh** method in multi-user environments in which the database schema is subject to change to retrieve current table definitions. Using the **Refresh** method on an **rdoTables** collection fetches table names from the base tables in the database. Using **Refresh** on a specific **rdoTable** object's **rdoColumns** collection fetches the table structures including column names and data types from the base tables.

Note Before you can use the name of a base table in the ***name*** argument of the **OpenResultset** method, you must first use the **Refresh** method against the **rdoTables** collection to populate it. You can also populate the **rdoTables** collection by referencing one of its members by its ordinal number. For example, referencing **rdoTables**(0) will populate the entire collection.

Refresh Method Example

The following example illustrates use of the **Refresh** method to rebuild an **rdoResultset** on the **RemoteData** control. The example resets the SQL property with a new query built using the concatenation technique. When the **Refresh** method is executed, the query is re-executed. Since the **Connect** property is not changed for each invocation of the Search procedure, the connection is not re-established each time – it is opened only on the first invocation. When the **Refresh** method is complete, the bound controls reflect data from the columns returned by the query.

```
Option Explicit
Private Sub Search_Click()
On Error GoTo eh
With MSRDC1
    .Connect = "UID=;PWD=;Database=Pubs;"
    .DataSourceName = "WorkDB"
    .SQL = "Select Au_Fname " _
       & " From Authors " _
       & " Where Au_Lname like '%" _
       & AuthorWanted & "%'"
    Debug.Print .SQL
    .Refresh

    If .Resultset.EOF Then
       MsgBox "No authors on file with that last name"
    End If
End With
Exit Sub

eh:
Dim er As rdoError
For Each er In rdoErrors
Debug.Print er
Next
Resume Next
End Sub
```

Requery Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthRequeryC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthRequeryX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthRequeryA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthRequeryS"}

Updates the data in an **rdoResultset** object by re-executing the query on which the object is based.

Syntax

object.**Requery** [*options*]

The **Requery** method syntax has these parts:

Part	Description
object	The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.
options	A **Variant** or constant that determines how the query is run, as specified in Settings.

Settings

You can use the following constant for the *options* part:

Constant	Value	Description
rdAsyncEnable	32	Execute operation asynchronously.

Remarks

Use this method to ensure an **rdoResultset** contains the most recent data. When you use **Requery**, all changes made to the data in the underlying table(s) by you and other users is displayed in the **rdoResultset**, and the first row in the **rdoResultset** becomes the current row.

If you use the **rdAsyncEnable** option, control returns to your application as soon as the query is begun, often before a result set is available. To test for completion of the query, use the **StillExecuting** property. The **rdoResultset** object is not valid until **StillExecuting** returns **False**. You can also use the QueryCompleted event to indicate when the query is completed.

If the **rdoParameter** objects have changed, their new values are used in the query used to generate the new **rdoResultset**.

Once the **Requery** method has been executed, all previously stored **rdoResultset** bookmarks are invalid.

If both the **BOF** and **EOF** property settings of the **rdoResultset** object are **True**, or the **RowCount** property is set to 0 after you use the **Requery** method, the query didn't return any rows and the **rdoResultset** contains no data.

You can't use the **Requery** method on **rdoResultset** objects whose **Restartable** property is set to **False**.

Requery Method Example

The following example illustrates use of the **Requery** method to re-execute an **rdoQuery**. Note that the **rdoResultset** is created only at form load and only re-executed on each invocation of the **Requery** method.

```
Option Explicit
Dim Cn As New rdoConnection
Dim Rs As rdoResultset
Dim Col As rdoColumn
Dim Qy As rdoQuery
Dim SQL As String
Dim TimeExpected As Single
Dim Ts As Single, Tn As Single


Private Sub SpWho_Click()
Rs.Cancel
With Rs
   .Requery
   While .StillExecuting
      SpinGlobe
      DoEvents
   Wend
   ShowRS
End With

End Sub
Sub ShowRS()
With Rs
   Form1.Cls
   For Each Col In .rdoColumns
      Print Col.Name,
   Next
   Print
   Do Until .EOF
      For Each Col In .rdoColumns
         Print Col,
      Next
      Print
      .MoveNext
   Loop
End With
End Sub
Sub SpinGlobe()
' Animate a globe here to show query is in progress.
Print ".";
End Sub

Private Sub Form_Load()
With Cn
   .Connect = "UID=;PWD=;Database=WorkDB;" _
   & "Server=sequel;Driver={SQL Server}" _
   & "DSN='';"
   .LoginTimeout = 5
   .EstablishConnection rdDriverNoPrompt, True
   Set Qy = .CreateQuery("SpWho", _
```

```
    "{ call master..sp_who (?) }")
    Qy.RowsetSize = 1
    Set Rs = Qy.OpenResultset(rdOpenForwardOnly, _
    rdConcurReadOnly, rdAsyncEnable)
    Show
    ShowRS
End With
End Sub
```

Update Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthUpdateS"}

Saves the contents of the copy buffer row to a specified updatable **rdoResultset** object and discards the copy buffer.

Syntax

object.**Update**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use **Update** to save the current row and any changes you've made to it to the underlying database table(s). Changes you make to the **rdoResultset** after using the **AddNew** or **Edit** methods can be lost if you do not use the **Update** method before the application ends.

Note When you use the ClientBatch cursor library, all updates to the base tables are deferred until you use the **BatchUpdate** method. In this case, the **Update** method updates the local **rdoResultset**, but does not update the base tables. These changes can be lost if the application ends before the **BatchUpdate** method has been completed.

Changes to the current row are lost if:

- You use the **Edit** or **AddNew** method, and then reposition the current row pointer to another row without first using **Update**.
- You use **Edit** or **AddNew**, and then use **Edit** or **AddNew** again without first using **Update**.
- You cancel the update with the **CancelUpdate** method.
- You set the **Bookmark** property to another row.
- You close the result set referred to by *object* without first using **Update**.
- The application ends before the **Update** method is executed, as when system power is interrupted.

To edit a row, use the **Edit** method to copy the contents of the current row to the copy buffer. If you don't use **AddNew** or **Edit** first, an error occurs when you use **Update** or attempt to add a new row.

To add a new row, use the **AddNew** method to initialize and activate the copy buffer.

Using **Update** produces an error under any of the following conditions:

- There is no current row.
- The connection or **rdoResultset** is read-only.
- No columns in the row are updatable.
- You do not have an **Edit** or **AddNew** operation pending.
- Another user has locked the row or data page containing your row.
- The user does not have permission to perform the operation.
- Depending on the driver and type of cursor being used, you might not be able to use the cursor to update the primary key.

UpdateControls Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthUpdateControlsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthUpdateControlsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthUpdateControlsS"}

Gets the current row from a **RemoteData control's** **rdoResultset** object and displays the appropriate data in controls bound to a **RemoteData** control.

Syntax

object.**UpdateControls**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use this method to restore the contents of bound controls to their original values, as when a user makes changes to data and then decides to cancel the changes.

This method creates the same effect as making the current row current again, except that no events occur. By not invoking any events, this method can be used to simplify an update operation because no additional validation or change event procedures are triggered.

UpdateRow Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthUpdateRowC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthUpdateRowA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthUpdateRowS"}

Saves the current values of bound controls to the database.

Syntax

object.**UpdateRow**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use this method to save the current contents of bound controls to the database during the Validate event, but without triggering the Validate event again. You can use this method to avoid triggering the Validate event. Using this method avoids a cascading event.

The **UpdateRow** method has the same effect as executing the **Edit** method, changing a column, and then executing the **Update** method, except that no events occur.

Note When you use the ClientBatch cursor library, all updates to the base tables are deferred until you use the **BatchUpdate** method. In this case, the **UpdateRow** method updates the local **rdoResultset**, but does not update the base tables. These changes can be lost if the application ends before the **BatchUpdate** method has been completed.

Whenever you attempt to update a row in the database, any validation rules must be satisfied before the row is written to the database. In the case of Microsoft SQL Server, these rules are established by Transact SQL defaults, rules, and triggers written to enforce referential and data integrity.

An update may not occur because of any of the following reasons, which can also trigger a trappable error:

- The page containing the row or the row itself is locked.
- The database or **rdoResultset** object isn't updatable.
- The user doesn't have permission to perform the operation.

Clear Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthClearC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthClearA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthClearS"}

Clears all members from the **rdoErrors** collection.

Syntax

object.**Clear**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

Use this method to remove all **rdoError** objects from the **rdoErrors** collection.

Generally, it is unnecessary to clear the **rdoErrors** collection because it is automatically cleared when the first error occurs after initiating a new operation. Use the **Clear** method in cases where you need to clear the **rdoErrors** collection manually, for example if you wish to clear errors that have already been processed.

BatchUpdate Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthBatchUpdateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthBatchUpdateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthBatchUpdateS"}

Performs a batched optimistic update.

Syntax

object.**BatchUpdate** (*SingleRow*, *Force*)

The **BatchUpdate** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
SingleRow	A **Boolean** value that is **True** if the update is done only for the current row, or **False** if the update applies to all rows in the batch. Default is False.
Force	A **Boolean** value that is **True** if the row or batch of rows will overwrite existing rows in the database regardless if they cause collisions or not. Default is **False**.

Remarks

This method performs a batch optimistic update operation. When using batch optimistic concurrency, it is necessary to call this method to actually send the changes back to the server.

Batch updates are used whenever you open a connection using the Client Batch cursor library (**rdUseClientBatch**). In this case, each time you use the **Update** or **UpdateRow** methods, the local **rdoResultset** is updated, but the base database tables are not changed. The **BatchUpdate** method is used to update the base database table(s) with any information changed since the **rdoResultset** was last created or synchronized with the **BatchUpdate** command.

The **BatchUpdate** method updates the **BatchCollisionRows** property to include a bookmark for each row that failed to update – collided with an existing row that has data more current than the **rdoResultset** object as it existed when first read. The **BatchCollisionCount** property indicates how many collisions occurred during the batch update process.

If you use the **CancelBatch** method, the changes saved to the local **rdoResultset** object are discarded. When you use the **CancelUpdate** method, only the current row's changes are rolled back to the state prior to execution of the last **Update** method.

The ***SingleRow*** parameter can be used in conjunction with the ***Force*** parameter to force the client's version of the data back into the database, even if collisions have occurred. The ***SingleRow*** parameter will tell RDO to only send the current row back to the server and not the entire batch, and the ***Force*** parameter will tell RDO to force the data in, and not use the normal optimistic concurrency detection.

Setting both the ***SingleRow*** and ***Force*** parameters to **True** overlays a single database row with the current updated **rdoResultset** row. This is useful when processing collision rows and you want to force your local version of the data to be saved regardless of the current database row setting.

Setting ***SingleRow*** to **False** and ***Force*** to **True** will cause all rows that are dirty to be forced into the database, which is useful as a shorthand way of forcing everything in (the last-one-in-wins scenario).

Setting ***SingleRow*** to **True** and ***Force*** to **False** will cause just the current row to go through the optimistic concurrency update, which is useful when you only want to update the current row, not the entire batch.

CancelBatch Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCancelBatchC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCancelBatchA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCancelBatchS"}

Cancels all uncommitted changes in the local cursor (used in batch mode)

Syntax

object.**CancelBatch**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **CancelBatch** method cancels all uncommitted changes in the local cursor, and reverts the data back to the state it was when originally fetched from the database. Note that this method does not refresh the data by re-querying the server like the **Refresh** method does — instead it just discards changes made in the local cursor that have not already been sent in a batch update operation.

When you use the **CancelUpdate** method, only the current row's changes are rolled back to the state prior to execution of the last **Update** method.

Batch updates are used whenever you open a connection using the Client Batch cursor library (**rdUseClientBatch**). In this case, each time you use the **Update** or **UpdateRow** methods, the local **rdoResultset** is updated, but the base database tables are not changed. The **BatchUpdate** method is used to update the base database table(s) with any information changed since the **rdoResultset** was last created or synchronized with the **BatchUpdate** command.

The **BatchUpdate** method updates the **BatchCollisionRows** property to include a bookmark for each row that failed to update – collided with an existing row that has data more current than the **rdoResultset** object as it existed when first read. The **BatchCollisionCount** property indicates how many collisions occurred during the batch update process.

CreateQuery Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthCreateQueryC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthCreateQueryX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthCreateQueryA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthCreateQueryS"}

Creates a new query object and adds it to the **rdoQueries** collection.

Syntax

object.**CreateQuery** *Name*, *SQLString*

The **CreateQuery** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an **rdoConnection** object
Name	Required. A string expression that evaluates to the name for the new object
SQLString	Optional. SQL query for the new prepared statement

Remarks

The **CreateQuery** method creates a new **rdoQuery** object for this connection and adds it to the **rdoQueries** collection. You can also declare stand-alone **rdoQuery** objects using the **Dim** statement as follows:

```
Dim myQuery as New rdoQuery
```

Stand-alone **rdoQuery** objects are not associated with a connection until you set the **ActiveConnection** property.

The **rdoQuery** corresponds to the ODBC prepared statement used to define a reusable SQL query that can contain parameters. You can execute the **rdoQuery** any number of times, and pass parameters that are substituted into the SQL statement before it is executed. Parameters are maintained in the **rdoParameters** collection. Generally, if you intend to execute a query more than once in your code, it is more efficient to use **rdoQuery** objects than to use the **Execute** or **OpenResultset** method on objects other than the **rdoQuery**.

The value passed for the ***Name*** parameter can be used with the **Item** method to locate the new object in its collection. If ***Name*** is not provided, the **rdoQuery** is appended to the **rdoQueries** collection, and the **rdoQuery** can be used by referencing the query variable or the **rdoQuery** object's ordinal value. If the object specified by name is already a member of the **rdoQueries** collection (including an empty string), a trappable error occurs. All **rdoQuery** objects are temporary — they are discarded when the **rdoConnection** object is closed.

To remove an **rdoQuery** object from an **rdoQueries** collection, use the **Close** method on the **rdoQuery**.

The ***SQLString*** parameter is optional, but if not provided, you must set the **SQL** property of the resulting **rdoQuery** object before executing it.

Use the **Execute** method to run an SQL statement in an **rdoQuery** object that does not return rows (an action query). Use the **OpenResultset** method to run an **rdoQuery** that returns rows.

If there is an unpopulated **rdoResultset** pending on a data source that can only support a single operation on an **rdoConnection** object, you cannot create additional **rdoQuery** or **rdoResultset** objects, or use the **Refresh** method on the **rdoTable** object until the **rdoResultset** is flushed, closed, or fully populated. For example, when using SQL Server 4.2 as a data source, you cannot create an additional **rdoResultset** object until you move to the last row of the current **rdoResultset** object. To populate the result set, use the **MoreResults** method to move through all pending result sets, or use

the **Cancel** or **Close** method on the **rdoResultset** to flush all pending result sets.

EstablishConnection Method (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthEstablishConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthEstablishConnectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthEstablishConnectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthEstablishConnectionS"}

Establishes a physical connection to an ODBC server.

Syntax

object.**EstablishConnection** *prompt*, *readonly*, *options*

The **EstablishConnection** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an **rdoConnection** object.
prompt	Optional. Integer value indicating ODBC prompting characteristic (see the **OpenConnection** method on the **rdoEnvironment** object).
readonly	Optional. Boolean value which is **True** if intending to use connection as read-only.
options	Optional. Integer value indicating connection options. This parameter has the same rules, restrictions and possible values that it does in the **OpenConnection** method of the **rdoEnvironment** object.

Remarks

This method causes the **rdoConnection** object to physically connect to the server, if it is not so already. This method is used when creating stand-alone **rdoConnection** objects or when re-connecting **rdoConnection** objects that have been disconnected using the **Close** method.

Unlike the **OpenConnection** method, the **EstablishConnection** method does *not* automatically append the **rdoConnection** object to the **rdoConnections** collection. If you want to add the newly established connection into the **rdoConnections** collection, you must use the **Add** method. You can use the **Remove** method to remove a member from the **rdoConnections** collection.

When using the Client Batch cursor library, the **EstablishConnection** method can be used to establish a connection once the **ActiveConnection** of an **rdoResultset** or **rdoQuery** object has been set to **Nothing**.

Just as with the **OpenConnection** method, the ***prompt*** argument dictates how the ODBC driver manager prompts the user for missing arguments needed to establish the connection. You can also request that the connection be made asynchronously by using the **rdAsyncEnable** option.

In general, you must set the **Connect** property and other appropriate properties of the **rdoConnection** object prior to making an attempt at connecting to a remote server.

See the **OpenConnection** method for details on how the **rdoConnection** properties should be set prior to attempting to use the **EstablishConnection** method.

GetClipString Method

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdmthGetClipStringC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdmthGetClipStringX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdmthGetClipStringA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdmthGetClipStringS"}

The **GetClipString** method returns a delimited string for '*n*' rows in a result set.

Syntax

ResulsetString = *object*.**GetClipString** (*NumRows*, [*ColumnDelimiter*],[*RowDelimiter*], *NullExpr*])

The **GetClipString** method syntax has these parts:

Part	Description
ResultsetString	A variable used to reference the entire result set as a delimited string.
Object	An object expression that evaluates to an **rdoResultset** object.
NumRows	Required: **Long** value. Number of rows to copy into the clip string.
ColumnDelimiter	Optional: **Variant**(**String)** expression used to separate data columns as described in Settings. Default is Tab (**VbTab**).
RowDelimiter	Optional: **Variant**(**String)**expression used to separate data rows as described in Settings. Default is carriage return (**VbCr**).
NullExpr	Optional: **Variant**(**String)**expression used when NULL values are encountered as described in Settings. Default is an empty string.

Settings

The row and column delimiters can be any length, but are generally one or two bytes long. Generally, the ***ResultsetString*** delimiters are determined by the **Clip** property of the target object. For example, if the string is applied to a grid control, columns are separated by tabs and the rows are separated by carriage returns (the default settings).

The ***NullExpr*** is used to substitute a suitable value in place of NULL values returned from the query. Generally, an empty string or “<null>” is used.

Remarks

The **GetClipString** method returns a delimited string for '*n'* rows in a result set based on the ***NumRows*** argument. If more rows are requested than are available, only the available rows are returned. Use the **RowCount** property to determine how many rows are actually fetched. The number of rows that can be fetched is constrained by available memory and should be chosen to suit your application. Don't expect to use **GetClipString** to bring your entire table or result set into memory if it is a large table.

Generally, **GetClipString** works just like the **GetRows** method except that the data is returned as a string instead of a 2-dimensional variant array. **GetClipString** can be used fill a grid control, or any control that has a **Clip** property. It can also be used to format export data from a result set to a sequential file.

After a call to **GetClipString**, the current row is positioned at the next unread row. That is, **GetClipString** is equivalent to using the **Move** (rows) method.

If you are trying to fetch all the rows using multiple **GetClipString** calls, use the **EOF** property to

determine if there are rows available. **GetClipString** returns less than the number requested either at the end of the **rdoResultset**, or if it cannot fetch a row in the range requested. For example, if a fifth row cannot be retrieved in a group of ten rows that you're trying to fetch, **GetClipString** returns four rows and leaves currency on the row that caused the problem. It will not generate a run-time error.

The ***ColumnDelimiter*** optional parameter can be used to substitute a different column delimiter than the default tab (**Chr**$(9)) character, and the ***RowDelimiter*** optional parameter can be used to substitute a different row delimiter. This is useful when working with a control that accepts a clip format, but requires different characters for the column and row delimiters (some grids have been known to require both a carriage return and a line feed character for a row delimiter).

GetClipString Example (Remote Data)

The following example creates a clip string from a result set containing a selected set of rows from the Publishers table and fills a **Grid** control from the string by applying it to the **Clip** property.

```
Dim rs As rdoResultset
Set rs = MyConnection.OpenResultset( _
"Select * from Publishers Where State = 'WA'", _
   rdOpenNone)
MyGrid.Rows = rs.RowCount
MyGrid.Cols = rs.rdoColumns.Count
MyGrid.SelStartRow = 1
MyGrid.SelEndRow = MyGrid.Rows
MyGrid.SelStartCol = 0
MyGrid.SelEndCol = MyGrid.Cols - 1
MyGrid.Clip = rs.GetClipString(rs.RowCount)
```

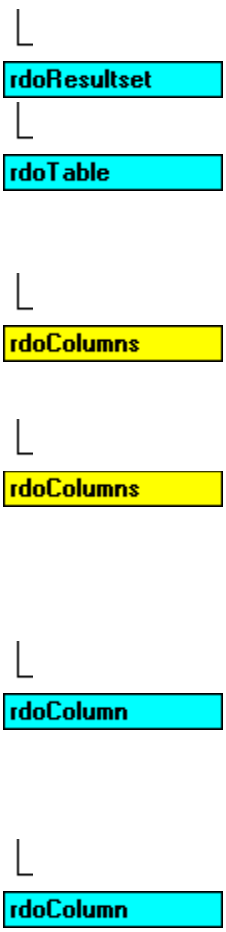
result set

The results of a query. Result sets might contain rows when a query contains a SELECT statement. Action queries do not return rows but do return result sets that contain information about the operation, such as rows affected.

rdoColumn Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoColumnC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoColumnX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdocolumnP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoColumnM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoColumnE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoColumnS"}

An **rdoColumn** object represents a column of data with a common data type and a common set of properties.



Remarks

The **rdoTable**, or **rdoResultset** object's **rdoColumns** collection represents the **rdoColumn** object in a row of data. You can use the **rdoColumn** object in an **rdoResultset** to read and set values for the data columns in the current row of the object. However, in most cases, references to the **rdoColumn** object is only implied because the **rdoColumns** collection is the **rdoResultset** object's default collection.

An **rdoColumn** object's name is determined by the name used to define the column in the data source table or by the name assigned to it in an SQL query. For example, if an SQL query aliases the column, this name is assigned to the **Name** property; otherwise, the column's name is used.

You manipulate database columns using an **rdoColumn** object and its methods and properties. For example, you can:

- Use the **Value** property of an **rdoColumn** to extract data from a specified column.
- Use the **Type** and **Size** property settings to determine the data type and size of the data.

- Use the **Updatable** property to see if the column can be changed.
- Use the **SourceColumn** and **SourceTable** property settings to locate the original source of the data.
- Use the **OrdinalPosition** property to get presentation order of the **rdoColumn** objects in an **rdoColumns** collection.
- Use the **Attributes** and **Required** property settings to determine optional characteristics and if Nulls are permitted in the column.
- Use the **AllowZeroLength** property to determine how zero-length strings are handled.
- Use the **BatchConflictValue,** and **OriginalValue** properties to resolve optimistic batch update conflicts.
- Use the **KeyColumn** to determine if this column is part of the primary key.
- Use the **Status** property to determine if the column has been modified.
- Use the **AppendChunk, ColumnSize**, and **GetChunk** methods to manipulate columns that require the use of these methods, as determined by the **ChunkRequired** property.

When you need to reference data from an **rdoResultset** column, you can refer to the **Value** property of an **rdoColumn** object by:

- Referencing the **Name** property setting using this syntax:

```
' Refers to the Au_Fname column rdoColumns("Au_Fname")
rs.rdoColumns("Au_Fname")
```

  -Or-

```
' Refers to the Au_Fname column
rs.rdoColumns!Au_Lname
```

- Referencing its ordinal position in the **rdoColumns** collection using this syntax:

```
rs.rdoColumns(0)
```

The **rdoTable** object's **rdoColumns** collection contains specifications for the data columns. You can use the **rdoColumn** object of an **rdoTable** object to map a base table's column structure. However, you cannot directly alter the structure of a database table using RDO properties and methods. You can, however, use data definition language (DDL) action queries to modify database schema.

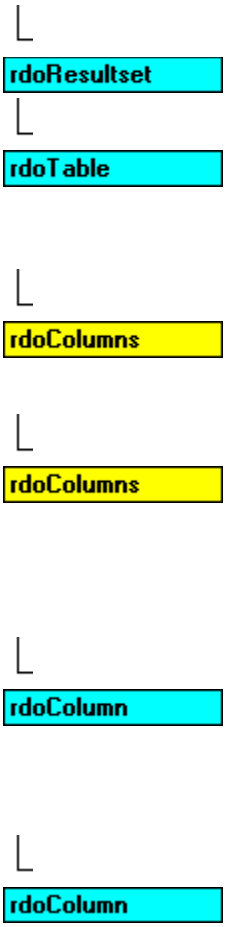
When the **rdoColumn** object is accessed as part of an **rdoResultset** object, data from the current row is visible in the **rdoColumn** object's **Value** property. To manipulate data in the **rdoResultset**, you don't usually reference the **rdoColumns** collection directly. Instead, use syntax that references the **rdoColumns** collection as the default collection of the **rdoResultset**.

```
dim rs As rdoResultset
Set rs = cn.OpenResultset("Select * from Authors" _
   & "Where Au_Lname = 'White'",rdOpenForwardOnly)
debug.print rs!Au_Fname
   'Refers to rdoRecordset object's rdoColumns collection.
```

rdoColumns Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoColumnSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoColumnSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoColumnsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoColumnsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoColumnsCollectionS"}

An **rdoColumns** collection contains all **rdoColumn** objects of an **rdoResultset**, or **rdoTable** object.



Remarks

The **rdoTable**, or **rdoResultset** object's **rdoColumns** collection represents the **rdoColumn** objects in a row of data. You use the **rdoColumn** object in an **rdoResultset** to read and set values for the data columns in the current row of the object.

The **rdoColumn** object is either created automatically by RDO when

- An **rdoTable**, or **rdoResultset** object is created.
- An **rdoTable** object is referenced.
- An **rdoResultset** is created via OpenResultset.

rdoColumn Object, rdoColumns Collection Example

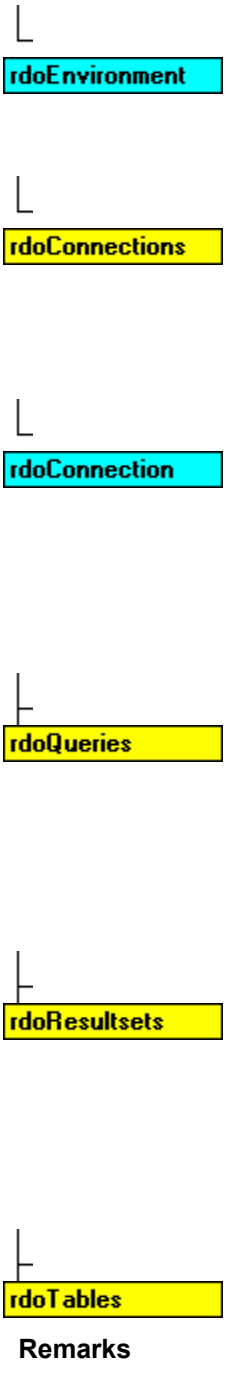
The following example opens a connection against an SQL Server database and creates an **rdoResultset** that returns two columns: one normal column, and one derived from an expression. Next, the example maps the **rdoColumn** objects returned from the result set.

```
Private Sub rdoColumnButton_Click()
Dim cl As rdoColumn
Dim rs As rdoResultset
Dim sSQL As String
Dim cn As rdoConnection
Dim connect As String

connect = "uid=;pwd=;database=pubs;"

Set cn = rdoEnvironments(0).OpenConnection(workdb, _
            rdDriverNoPrompt, False, connect)

sSQL = "Select Pub_ID, Max(Price) BestPrice " _
    & " from Titles Group by Pub_ID"

Set rs = cn.OpenResultset(sSQL, rdOpenForwardOnly, _
    rdConcurReadOnly)

With rs
    For Each cl In .rdoColumns
        Print cl.Name; "-"; cl.Type; ":"; cl.Size, _
         cl.SourceTable, cl.SourceColumn
    Next cl
    Print
    Do Until .EOF
        For Each cl In .rdoColumns
            Print cl.Value,
        Next cl
        Print
        .MoveNext
    Loop
End With
End Sub
```

rdoConnection Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoConnectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoConnectionP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoConnectionM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoConnectionE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoConnectionS"}

An **rdoConnection** object represents an open connection to a remote data source and a specific database on that data source, or an allocated but as yet unconnected object, which can be used to subsequently establish a connection.



Remarks

Generally, an **rdoConnection** object represents a physical connection to the remote data source and corresponds to a single ODBC **hDbc** handle. A connection to a remote data source is required before you can access its data. You can open connections to remote ODBC data sources and create **rdoConnection** objects with either the **RemoteData** control or the **OpenConnection** method of an **rdoEnvironment** object.

To establish a connection to a remote server using the **rdoConnection** object, you can use the **OpenConnection** method to gather the ***connect***, ***dsname***, ***readonly*** and ***prompt*** arguments and open the connection. These arguments are then applied to the newly created **rdoConnection** object. You can also establish connections using the **RemoteData** control.

Creating Stand Alone rdoConnection Objects

You can also create a new **rdoConnection** object that is *not* immediately linked with a specific physical connection to a data source. For example, the following code creates a new stand-alone **rdoConnection** object:

```
Dim X as new rdoConnection.
```

Once created, you can set the properties of a stand-alone **rdoConnection** object and subsequently use the **EstablishConnection** method. This method determines how users are prompted — based on the ***prompt*** argument, and sets the read-only status of the connection based on the ***readonly*** argument.

When using this technique, RDO sets the following properties based on **rdoEngine** default values: **CursorDriver**, **LoginTimeout**, **UserName**, **Password** and **ErrorThreshold**. The **CursorDriver** and **LoginTimeout** properties can be set in the **rdoConnection** object itself and the **UserName** and **Password** can be set through arguments in the connect string. Once the connection is open, all of these properties are read-only.

When you declare a stand-alone **rdoConnection** object or use the **EstablishConnection** method, the object is not automatically appended to the **rdoConnections** collection. Use the **Add** or **Remove** methods to add or delete stand-alone **rdoConnection** objects to or from the **rdoConnections** collection. It is not necessary, however to add an **rdoConnection** object to the **rdoConnections** collection before it can be used to establish a connection.

Note RDO 1.0 collections behave differently than Data Access Object (DAO) collections. When you **Set** a variable containing a reference to a RDO object like **rdoResultset**, the existing **rdoResultset** is *not* closed and removed from the **rdoResultsets** collection. The existing object remains open and a member of its respective collection.

In contrast, RDO 2.0 collections do not behave in this manner. When you use the Set statement to assign a variable containing a reference to an RDO object, the existing object *is* closed and removed from the associated collection. This change is designed to make RDO more compatible with DAO.

Asynchronous Operations

Both the **EstablishConnection** and **OpenConnection** methods support synchronous, asynchronous, and event-managed operations. By setting the **rdAsyncEnable** option, control returns to your application *before* the connection is established. Once the **StillConnecting** property returns **False**, and the **Connect** event fires, the connection has either been made or failed to complete. You can check the success or failure of this operation by examining errors returned through the **rdoErrors** collection.

Opening Connections without Data Source Names

In many situations, it is difficult to ensure that a registered Data Source Name (DSN) exists on the target system, and in some cases it is not advisable to create one. Actually, a DSN is not needed to establish a connection if you are using the default network protocol (named pipes) and you know the name of the server and ODBC driver. If this is the case, you can establish a *DSN-less* connection by following these steps:

1. Set the ***DSN*** argument of the connect string to an empty string (DSN='').
2. Include the server name in the connect string.
3. Include the ODBC driver name in the connect string. Since many driver names have more than one word, enclose the name in curly braces { }.

Note This option is not available if you need to use other than the named pipes network protocol or one of the other DSN-set options such as OEMTOANSI conversion.

For example, the following code opens a read-only ODBC cursor connection against the SQL Server "SEQUEL" and includes a simple error handler:

```
Sub MakeConnection()
Dim rdoCn As New rdoConnection
On Error GoTo CnEh
With rdoCn
    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=SEQUEL;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .CursorDriver = rdUseODBC
    .EstablishConnection rdDriverNoPrompt, True
End With
Exit Sub
CnEh:
Dim er As rdoError
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description, er.Number
    Next er
    Resume Next
End Sub
```

Choosing a Specific Database

Once a connection is established, you can manipulate a database associated with the **rdoConnection** using the **rdoConnection** object and its methods and properties. For servers that support more than one database per connection, the default database is:

- Assigned to the user name by the database system administrator
- Specified with the DATABASE connect argument used when the **rdoConnection** is created.
- Specified in the registered ODBC data source entry.
- Selected by using an SQL statement such as USE <database> submitted with an action query.

All queries executed against the server assume this default database unless another database is specifically referenced in your SQL query.

Preparing for Errors when Connecting

There are a variety of reasons why you might be unable to connect to your remote database. Consider the following conditions that can typically prevent connections from completing:

- Your server might not have sufficient connection resources due to administrative settings or licensing restrictions.
- Your user might not have permission to access the network, server, or database with the password provided.
- The server, network or WAN bridges might be down or simply running slower than expected.

Closing the rdoConnection

When you use the **Close** method against an **rdoConnection** object, any open **rdoResultset,** or

rdoQuery objects are closed. However, if the **rdoConnection** object simply loses scope, these objects remain open until the **rdoConnection** or the objects are explicitly closed. Closing a connection is not recommended when there are incomplete queries or uncommitted transactions pending.

Closing a connection also removes it from the **rdoConnections** collection. However, the **rdoConnection** object itself is not destroyed. If needed, you can use the **EstablishConnection** method to re-connect to the same server using the same settings, or change the **rdoConnection** object's properties and then use **EstablishConnection** to connect to another server.

Closing a connection also instructs the remote server to discard any instance-specific objects associated with the connection. For example, server-side cursors, temporary tables or any other objects created in the *TempDB* database on SQL Server are all dropped.

Working with rdoConnection Methods and Properties

You can manipulate the connection, databases, and queries associated with them using the methods and properties of the **rdoConnection** object. For example, you can:

- Use the **CursorDriver** property to determine the type of cursor requested by result sets created against the connection.
- Use the **OpenResultset** method to create a new **rdoResultset** object.
- Use the **LastQueryResults** to reference the last **rdoResultset** created against this connection.
- Use the **QueryTimeout** or **LoginTimeout** properties to specify how long the ODBC driver manager should wait before abandoning a query or connection attempt.
- Use the **RowsAffected** property to determine how many rows were affected by the last action query.
- Use the **Execute** method to run an action query or pass an SQL statement to a database for execution.
- Use the **CreateQuery** method to create a new **rdoQuery** object.
- Use the **Close** method to close an open connection, remove the **rdoConnection** object from the **rdoConnections** collection, deallocate the connection handle, and terminate the connection.
- Use the **Transactions** property to determine if the connection supports transactions, which you can implement using the **BeginTrans**, **CommitTrans**, and **RollbackTrans** methods.
- Use the **AsyncCheckInterval** property to determine how often RDO should poll for a completed asynchronous operation.
- Use the ODBC API with the **hDbc** property to set connection options.
- Use the **Connect** property to determine the ***connect*** argument used in the **OpenConnection** method, or the **Connect** property of the **RemoteData** control.

rdoConnection Events

The following events are fired as the **rdoConnection** object is manipulated. These can be used to micro-manage the process of connecting and disconnecting and provide additional retry handling in query timeout situations.

Event Name	Description
BeforeConnect	Fired before ODBC is called to establish the connection.
Connect	Fired after a connection is established.
Disconnect	Fired after a connection has been closed
QueryComplete	Fired after a query run against this connection is complete
QueryTimeout	Fired after the QueryTimeout period is exhausted.

Addressing the rdoConnection Object

The **Name** property setting of an **rdoConnection** specifies the data source name (DSN) parameter used to open the connection. This property is often empty as it is not used when making a DSN-less connection. In cases where you specify a different DSN to open each connection, you can refer to any **rdoConnection** object by its **Name** property setting using the following syntax. This code Refers to the connection opened against the *Accounting* DSN:

```
rdoConnections("Accounting")
```

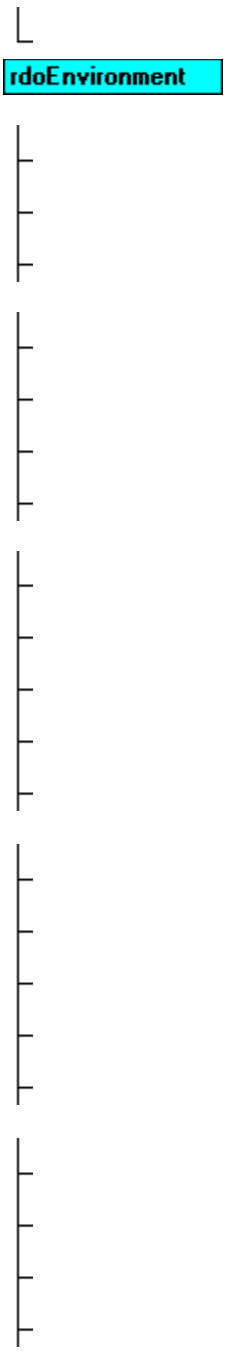
You can also refer to the object by its ordinal number using this syntax (which refers to the first member of the **rdoConnections** collection):

```
rdoConnections(0)
```

rdoConnections Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoConnectionSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoConnectionSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoConnectionsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoConnectionsM;vamthAdd;vamthRemove"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoConnectionSE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoConnectionsCollectionS"}

An **rdoConnections** collection contains all **rdoConnection** objects opened or created in an **rdoEnvironment** object of the remote database engine, or allocated and appended to the **rdoConnections** collection using the **Add** method.



Remarks

The **rdoConnections** collection is used to manage your **rdoConnection** objects. However, only **rdoConnection** objects created using the **OpenConnection** method, or using the **RemoteData** control are automatically appended to the collection. When you allocate a stand-alone **rdoConnection** object, it is not appended to the **rdoConnections** collection until you use the **Add** method.

Note RDO 1.0 collections behave differently than Data Access Object (DAO) collections. When you **Set** a variable containing a reference to a RDO object like **rdoResultset**, the existing **rdoResultset** is *not* closed and removed from the **rdoResultsets** collection. The existing object remains open and a member of its respective collection.

In contrast, RDO 2.0 collections do not behave in this manner. When you use the Set statement to assign a variable containing a reference to an RDO object, the existing object *is* closed and removed from the associated collection. This change is designed to make RDO more compatible with DAO.

Closing rdoConnection Objects

When you use the **Close** method against an **rdoConnection** object, any open **rdoResultset,** or **rdoQuery** objects are closed and the **rdoConnection** object is removed from the **rdoConnections** collection. However, if the **rdoConnection** object simply loses scope, these objects remain open until the **rdoConnection** or the objects are explicitly closed.

rdoConnection Object, rdoConnections Collection Example

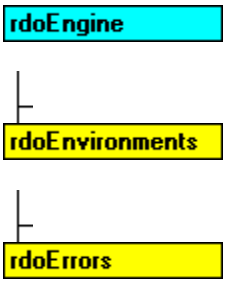
This example establishes a stand-alone **rdoConnection** object, sets its properties and uses the **EstablishConnection** method to open the connection.

```
Option Explicit
Dim cn As New rdoConnection

Private Sub rdoConnectionButton_Click()
WorkingLight.Caption = "Connecting"
With cn
    .Connect = "DSN=BadServerName;" _
     & "UID=;PWD=;DATABASE=WorkDB;"
    .LoginTimeout = 20
    .CursorDriver = rdUseOdbc
    .EstablishConnection _
      prompt:=rdDriverNoPrompt, _
      Option:=rdAsyncEnable
End With
While .StillConnecting
  ToggleLight     ' Flash status indicator
  DoEvents
Wend
WorkingLight = True     ' Show status indicator
WorkingLight.Caption = "Connected"
End Sub

Sub ToggleLight() 'Flashes "Opening" light
WorkingLight = Not WorkingLight
End Sub
```

rdoEngine Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoEngineC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoEngineX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoEngineP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoEngineM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoEngineE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoEngineS"}

The **rdoEngine** object represents the remote data source. As the top-level object, it contains all other objects in the hierarchy of Remote Data Objects (RDO).



Remarks

The **rdoEngine** object can represent a remote database engine or another data source managed by the ODBC driver manager as a database. The **rdoEngine** object is a predefined object, therefore you can't create additional **rdoEngine** objects and it isn't a member of any collection.

The **rdoEngine** object is used to reference the **rdoEnvironments** collection, or establish default values for newly created **rdoEnvironment** objects. When an **rdoEnvironment** object is created, its properties are initialized based on the default values set in the **rdoEngine**. A default **rdoEnvironments**(0) object is created automatically when it is first referenced.

The **rdoEngine** object fires the InfoMessage event when an informational message is returned from the remote data source. Informational messages are indicated by an ODBC SQL_SUCCESS_WITH_INFO return code. These messages are placed in the **rdoErrors** collection. In cases where several messages arrive at once, only a single InfoMessage event is fired — after the last message arrives and has been added to the **rdoErrors** collection.

Setting Default rdoEnvironment Properties

The following properties establish default settings for all newly-created **rdoEnvironment** objects. They are also used when instantiating stand-alone **rdoConnection** objects.

- Use the **rdoDefaultLoginTimeout** property to determine the **rdoEnvironment** object's default **LoginTimeout** property used in connection timeout management.
- Use the **rdoDefaultCursorDriver** property to determine the **rdoEnvironment** object's default **CursorDriver** value. This property determines if the ODBC driver manager creates client batch, local, server-side, or no cursors.
- Use the **rdoDefaultUser** and **rdoDefaultPassword** properties to determine the default **rdoEnvironment** object's **UserName** and **Password** properties. These determine the user name and password when opening connections if no specific values are supplied.

Working with other rdoEngine Properties and Methods

You can establish the default configuration of new **rdoEnvironment** objects and create new ODBC data source entries using the properties and methods of the **rdoEngine** object. For example, you can:

- Use the **rdoEnvironments** collection to examine **rdoEnvironment** objects that have been appended to the collection. Note that **rdoEnvironment** objects can be allocated as stand-alone objects.

- Use the **rdoLocaleID** property to determine which language-localized DLLs are loaded.
- Use the **Version** property to examine the version of RDO in use.
- Use the **rdoErrors** collection to examine information about errors generated by the ODBC interface. Errors generated by Visual Basic are maintained in a separate **Errors** collection.
- Use the **rdoRegisterDataSource** method to create a new data source entry in the Windows System Registry.
- Use the **rdoCreateEnvironment** method to create a new **rdoEnvironment** object. You can also allocate a new **rdoEnvironment** object by coding

```
Dim MyEnv as New rdoEnvironment
```

rdoEngine Object Example

This example sets a number of **rdoEngine** properties and creates a customized **rdoEnvironment** object based on these new default settings. Note that while your code can *set* a password in an **rdoEnvironment** object, it cannot be read once it is set.

```
Dim en As rdoEnvironment
Private Sub Form_Load()
With rdoEngine
    .rdoDefaultLoginTimeout = 20
    .rdoDefaultCursorDriver = rdUseOdbc
    .rdoDefaultUser = "Fred"
    .rdoDefaultPassword = ""
End With
Set en = rdoEnvironments(0)
'
'  Dump current rdoEnvironments collection
'  and display current properties where
'  possible.
'
For Each en In rdoEnvironments
    Debug.Print "LoginTimeout:" & en.LoginTimeout
    Debug.Print "CursorDriver:" & en.CursorDriver
    Debug.Print "User:" & en.UserName
   ' (Write-only) Debug.Print "Password:" & en.Password
Next
End Sub
```

rdoEnvironment Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoEnvironmentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoEnvironmentX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoEnvironmentP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoEnvironmentM;vamthAdd;vamthRemove"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoEnvironmentS"}

An **rdoEnvironment** object defines a logical set of connections and transaction scope for a particular user name. It contains both open and allocated but unopened connections, provides mechanisms for simultaneous transactions, and provides a security context for data manipulation language (DML) operations on the database.

Remarks

Generally, an **rdoEnvironment** object corresponds to an ODBC environment that can be referred to by the **rdoEnvironment** object's **hEnv** property. However, if the *Name* argument is *not* provided when the **rdoEnvironment** object is created by the **rdoCreateEnvironment** method, a stand-alone **rdoEnvironment** is created that is not added to the **rdoEnvironments** collection. Stand-alone **rdoEnvironment** objects are not exposed to other in-process DLLs unless specifically designated as public. If the reference count for any private **rdoEnvironment** is reduced to zero, all **rdoConnections** associated with the **rdoEnvironment** are closed.

Once you set the properties of an **rdoEnvironment** object, you can use the **Add** method to append it to the **rdoEnvironments** collection or the **Remove** method to detach and deallocate the object. The **Name** property is read-only and is determined by the specific remote data object.

The default **rdoEnvironment** is created automatically when the **RemoteData** control is initialized, or the first remote data object is referenced in code. The **Name** property of **rdoEnvironments**(0) is "Default_Environment". The user name and password for **rdoEnvironments**(0) are both "".

rdoEnvironment objects can be created with the **rdoCreateEnvironment** method of the **rdoEngine** object which automatically appends the new object to the **rdoEnvironments** collection. All **rdoEnvironment** objects created in this manner are assigned properties based on the default properties set in the **rdoEngine** object.

The user name and password information from the **rdoEnvironment** is used to establish the connection if these values are not supplied in the ***connect*** argument of the **OpenConnection** method, or in the **Connect** property of the **RemoteData** control.

All **rdoEnvironment** objects share a common **hEnv** value that is created on an application basis. Use

the **rdoEnvironment** object to manage the current ODBC environment, or to start an additional connection. In an **rdoEnvironment**, you can open multiple connections, manage transactions, and establish security based on user names and passwords. For example, you can:

- Create an **rdoEnvironment** object using the **Name**, **Password**, and **UserName** properties to establish a named, password-protected environment. The environment creates a scope in which you can open multiple connections and conduct one instance of coordinated transactions.
- Use the **CursorDriver** property to determine which cursor driver library is used to build **rdoResultset** objects. You can choose one of four types of cursors, or set the **CursorDriver** property to **rdUseNone** to indicate that no cursor is to be used to manage result sets.
- Use the **OpenConnection** method to open one or more existing connections in that **rdoEnvironment**.
- Use the **LoginTimeout** property to determine how long the ODBC drivers should wait before abandoning the connection attempt.
- Use the **BeginTrans**, **CommitTrans**, and **RollbackTrans** methods to manage transaction processing within an **rdoEnvironment** across several connections.
- Use several **rdoEnvironment** objects to conduct multiple, simultaneous, independent, and overlapping transactions.
- Use the **Close** method to terminate an environment and the connection and remove the **rdoEnvironment** object from the **rdoEnvironments** collection. This also closes all connections associated with the object.

Managing Transactions

The **rdoEnvironment** also determines transaction scope. Committing an **rdoEnvironment** transaction commits all open **rdoConnection** databases and their corresponding open **rdoResultset** objects. This does not imply a two-phase commit operation — simply that individual **rdoConnection** objects are instructed to commit any pending operations — one at a time.

For Microsoft SQL Server databases, the Distributed Transaction Coordinator (DTC) can be used to manage blocks of transactions simply by introducing the SQL query with the BEGIN DISTRIBUTED TRANSACTION statement. DTC facilitates the creation of network-wide database updates through its own two-phase commit protocol. Whenever SQL Server commits a transaction, the DTC ensures all related resources also commit the transaction. If any part of the transaction fails, the DTC ensures that the entire transaction is rolled back across all enlisted servers.

When you use transactions, all databases in the specified **rdoEnvironment** are affected – even if multiple **rdoConnection** objects are opened in the **rdoEnvironment**. For example, suppose you use a **BeginTrans** method against one of the databases visible from the connection, update several rows in the database, and then delete rows in another **rdoConnection** object's database. When you use the **RollbackTrans** method, both the update and delete operations are rolled back. To avoid this problem, you can create additional **rdoEnvironment** objects to manage transactions independently across **rdoConnection** objects. Note that transactions executed by multiple **rdoEnvironment** objects are serialized and are not atomic operations. Because of this, their success or failure is not interdependent. This is an example of batched transactions.

You can execute nested transactions *only* if your data source supports them. For example, on a single connection, you can execute a BEGIN TRANS SQL statement, execute several UPDATE queries, and another BEGIN TRANS statement. Any operations executed after the second BEGIN TRANS SQL statement can be rolled back independently of the statements executed after the first BEGIN TRANS. This is an example of nested transactions. To commit the first set of UPDATE statements, you must execute a COMMIT TRANS statement, or a ROLLBACK TRANS statement for each BEGIN TRANS executed.

rdoEnvironment Events

The following events are fired as the **rdoEnvironment** object is manipulated. These can be used to

micro-manage RDO transactions associated with the **rdoEnvironment** or to synchronize some other process with the transaction.

Event Name	Description
BeginTrans	Fired after the BeginTrans method has completed.
CommitTrans	Fired after the CommitTrans method has completed.
RollbackTrans	Fired after the RollbackTrans method has completed.

Addressing rdoEnvironment Objects

The **Name** property of **rdoEnvironment** objects is set from the *name* argument passed to the **rdoCreateEnvironment** method. You can refer to any other **rdoEnvironment** object by specifying its **Name** property setting using this syntax:

rdoEnvironments("MyEnvName")

or simply:

rdoEnvironments!*MyEnvName*

You can also refer to **rdoEnvironment** objects by their position in the **rdoEnvironments** collection using this syntax (where *n* is the *n*th member of the zero-based **rdoEnvironments** collection):

rdoEngine.rdoEnvironments(*n*)

or simply:

rdoEnvironments(*n*)

rdoEnvironments Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoEnvironmentSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoEnvironmentSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoEnvironmentsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoEnvironmentsM;vamthAdd;vamthRemove"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoEnvironmentsCollectionS"}

The **rdoEnvironments** collection contains all active **rdoEnvironment** objects of the **rdoEngine** object.

Remarks

rdoEnvironment objects are created with the **rdoCreateEnvironment** method of the **rdoEngine** object. Newly created **rdoEnvironment** objects are automatically appended to the **rdoEnvironments** collection unless you do not provide a name for the new object when using the **rdoCreateEnvironment** method or simply declare a new **rdoEnvironment** object in code.

The **rdoEnvironments** collection is automatically initialized with a default **rdoEnvironment** object based on the default properties set in the **rdoEngine** object.

If you use the **Close** method against an **rdoEnvironment** object, all **rdoConnections** it contains are closed and the object is removed from the **rdoEnvironments** collection.

rdoEnvironment Object, rdoEnvironments Collection Example

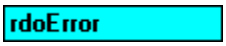
The following example illustrates creation of the **rdoEnvironment** object and its subsequent use to open an **rdoConnection** object.

```
Private Sub rdoEnvironmentButton_Click()
Dim en As rdoEnvironment
Dim cn As rdoConnection
Set en = rdoEngine.rdoEvironments(0)
With en
    en.CursorDriver = rdUseOdbc
    en.LoginTimeout = 5
    en.Name = "TransOp1"
    Set cn = en.OpenConnection(dsname:="", _
        prompt:=rdDriverNoPrompt, _
        Connect:="UID=;PWD=;" _
        driver={SQL Server};Server=SEQUEL;", _
        Options:=rdAsyncEnable)
End With
Print "Connecting ";
While cn.StillConnecting
    Print ".";
    DoEvents
Wend
Print "done."

End Sub
```

rdoError Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoErrorC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoErrorP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":""} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoErrorS"}

Contains details about remote data access errors.



Remarks

Any operation involving remote data objects can potentially generate one or more ODBC errors or informational messages. As each error occurs or as messages are generated, one or more **rdoError** objects are placed in the **rdoErrors** collection of the **rdoEngine** object. When a subsequent RDO operation generates an error, the **rdoErrors** collection is cleared, and the new set of **rdoError** objects is placed in the **rdoErrors** collection. RDO operations that don't generate an error have no effect on the **rdoErrors** collection. To make error handling easier. you can use the **Clear** method to purge the **rdoErrors** collection between operations.

Generally, all ODBC errors generate a trappable Visual Basic error of some kind. This is your cue to check the contents of the **rdoErrors** collection for any and all errors resulting from the last operation which provide specific details on the cause of the error.

Not all errors generated by ODBC are fatal. In the normal course of working with connections, default databases, stored procedure print statements and other operations, the remote server often returns warnings or messages that are usually safe to ignore. When an informational message arrives, the **rdoEngine** InfoMessage event is fired. You should examine the **rdoErrors** collection in this event procedure.

If the severity of the error number is below the error threshold as specified in either the **rdoDefaultErrorThreshold** or **ErrorThreshold** property, then a trappable error is triggered when the error is detected. Otherwise, an **rdoError** object is simply appended to the **rdoErrors** collection. To control trappable errors in Microsoft SQL Server, you should use the Transact SQL RAISERROR statement coupled with an appropriate *Severity* argument to indicate the error or other information.

Use the **rdoError** object to determine the type and severity of any errors generated by the RemoteData control or RDO operations. For example, you can:

- Use the **Description** property to display a text message describing the error.
- Use the **Number** property to determine the native data source error number.
- Use the **Source** property to determine the source of the error and the object class causing the error.
- Use the **SQLRetCode** and **SQLState** properties to determine the ODBC return code and **SQLState** flags.
- Use the **Clear** method on the **rdoErrors** collection to remove all **rdoError** objects. In most cases, it is not necessary to use the **Clear** method because the **rdoErrors** collection is cleared automatically when a new error occurs.

Members of the **rdoErrors** collection aren't appended as is typical with other collections. The most general errors are placed at the end of the collection (**Count** -1), and the most detailed errors are placed at index 0. Because of this implementation, you can often determine the root cause of the failure by examining **rdoErrors**(0).

The set of **rdoError** objects in the **rdoErrors** collection describes one error. The first **rdoError** object is the lowest level error, the second is the next higher level, and so forth. For example, if an ODBC error occurs while the **RemoteData** control tries to create an **rdoResultset** object, the last **rdoError** object contains the RDO error indicating the object couldn't be opened. The first error object contains the lowest level ODBC error. Subsequent errors contain the ODBC errors returned by the various layers of ODBC. In this case, the driver manager, and possibly the driver itself, returns separate errors which generate **rdoError** objects.

The **rdoErrors** collection is also used to manage informational messages returned by the data source. For example, messages returned back from PRINT statements, showplan requests, or DBCC operations in SQL Server are returned as **rdoError** objects in the **rdoErrors** collection. This type of message causes the InfoMessage event to fire, but does not trip a trappable error. Because of this, you must check the **rdoErrors** collection's **Count** property to see if any new errors have arrived.

rdoErrors Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoErrorSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoErrorSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoErrorsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoErrorsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoErrorsCollectionS"}

Contains all stored **rdoError** objects which pertain to a single operation involving Remote Data Objects (RDO).

rdoError

Remarks

Any operation involving remote data objects can generate one or more errors. As each error occurs, one or more **rdoError** objects are placed in the **rdoErrors** collection of the **rdoEngine** object. When another RDO operation generates an error, the **rdoErrors** collection is cleared, and the new set of **rdoError** objects is placed in the **rdoErrors** collection. RDO operations that don't generate an error have no effect on the **rdoErrors** collection.

- Use the **Clear** method on the **rdoErrors** collection to remove all **rdoError** objects. In most cases, it is not necessary to use the **Clear** method because the **rdoErrors** collection is cleared automatically when a new error occurs.

Members of the **rdoErrors** collection aren't appended as is typical with other collections. The most general errors are placed at the end of the collection (**Count** -1), and the most detailed errors are placed at index 0. Because of this implementation, you can determine the root cause of the failure by examining **rdoErrors**(0).

The set of **rdoError** objects in the **rdoErrors** collection describes one error. The first **rdoError** object is the lowest level error, the second is the next higher level, and so forth. For example, if an ODBC error occurs while the **RemoteData** control tries to create an **rdoResultset** object, the last **rdoError** object contains the RDO error indicating the object couldn't be opened. The first error object contains the lowest level ODBC error. Subsequent errors contain the ODBC errors returned by the various layers of ODBC. In this case, the driver manager, and possibly the driver itself, returns separate errors which generate **rdoError** objects.

rdoError Object, rdoErrors Collection Example

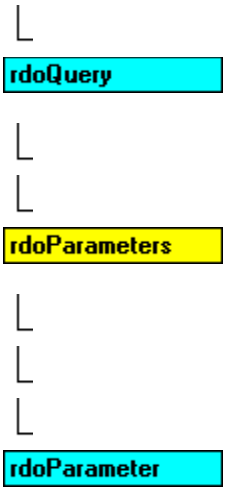
The following code illustrates a simple design-time RDO error handler. Note that the handler simply displays the errors in the **rdoErrors** collection in the Immediate window.

```
Dim er as rdoError
On Error GoTo CnEh
.
.
.

CnEh:
Dim er As rdoError
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description, er.Number
    Next er
    Resume Next
```

rdoParameter Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoParameterC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoParameterX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoParameterP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdobjrdoParameterM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoParameterS"}

An **rdoParameter** object represents a parameter associated with an **rdoQuery** object.



Remarks

When working with stored procedures or SQL queries that require use of arguments that change from execution to execution, you should create an **rdoQuery** object to manage the query and its parameters. For example, if you submit a query that includes information provided by the user such as a date range, or part number, RDO and the ODBC interface can insert these values automatically into the SQL statement at specific positions in the query.

Providing Parameters

Your query's parameters can be provided in a number of ways:

- As hard-coded arguments in the SQL query string.

```
"Select Name from Animals Where ID = 'Cat'"
```

- As concatenated text or numeric values extracted from **TextBox**, **Label** or other controls.

```
"Select Name from Animals Where ID = '"  _
      & IDWanted.Text & "'"
```

- As the question mark (?) parameter placeholders.

```
"Select Name from Animals Where ID = ?"
```

- As the question mark (?) parameter placeholders in a stored procedure call that accepts input, output and/or return status arguments.

```
"{ ? = Call MySP (?, ?, ?) }"
```

Note Stored procedure invocations that use the Call syntax (as shown above) are executed in their "native" format so they do not require parsing and data conversion by the ODBC Driver Manager. Because of this the Call syntax can be executed somewhat faster than other syntaxes.

Using Parameter Markers

The only time you *must* use parameter markers is when executing stored procedures that require input, output or return status arguments. If the stored procedure only requires input arguments, these can be provided in-line as imbedded values concatenated into the query (as shown below).

When the **rdoParameter** collection is first referenced (but not before) RDO and the ODBC interface pre-processes the query, and creates an **rdoParameter** object for each *marked* parameter. You can also create queries with multiple parameters, and in this case you can mark some parameters and provide the others by hard-coding or concatenation – in any combination. However, all marked parameters must appear to the left of all other parameters. If you don't, a trappable error occurs indicating "Wrong number of parameters".

Note Due to the extra overhead involved in creating and managing **rdoQuery** objects and their **rdoParameters** collection, you should not use parameter queries for SQL statements that do not change from execution to execution — especially those that are executed only once or infrequently.

Marking Parameters

Each query parameter that you want to have RDO manage must be indicated by a question mark (?) in the text of the SQL statement, and correspond to an **rdoParameter** object referenced by its ordinal number counting from zero – left to right. For example, to execute a query that takes a single input parameter, your SQL statement would look something like this:

```
SQL$ = "Select Au_Lname, Au_Fname where Au_ID Like ? "
Dim qd as rdoQuery, rd as rdoResultset
Set qd = CreateQuery ("SeekAUID", SQL$)
qd(0) = "236-66-%"
set rd = qd.OpenResultset(rdOpenForwardOnly)
```

Note You can also create an **rdoQuery** object using the Query Connection designer and name and set the data type and direction of individual parameters.

Acceptable Parameters

Not all types of data are passable as parameters. For example you cannot always use a TEXT or IMAGE data type as an OUTPUT parameter. In addition, if your query does not require parameters or has no parameters in a specific invocation of the query, you cannot use parenthesis in the query. For example, for a stored procedure that does not require parameters could be coded as follows:

```
"{ ? = Call MySP }"
```

When submitting queries that return output parameters, these parameters must be submitted at the end of the list of your query's parameters. While it is possible to provide both marked and unmarked (in-line) parameters, your output parameters must still appear at the end of the list of parameters.

All in-line parameters must be provided to the right of marked parameters. If this is not the case, RDO returns an error indicating "Wrong number of parameters".

RDO 2.0 supports BLOB data types as parameters and you also can use the **AppendChunk** method against the **rdoParameter** object to pass TEXT or IMAGE data types as parameters into a procedure.

Identifying the Parameter's Data Type

When your parameter query is processed by ODBC, it attempts to identify the data type of each parameter by executing ODBC functions that query the remote server for specific information about the query. In some cases, the data type cannot be correctly determined. In these cases, use the **Type** property to set the correct data type or create a custom query using the User Connection Designer.

For example, in the following query, the parameter passed to the TSQL **Charindex** function is typed as an integer. While this is correct for the function itself, the parameter is referencing a string argument of the TSQL function, so it must be set to an ODBC character type to work properly.

```
Dim SQL as string, qd as rdoQuery
SQL = "Select * From Titles " _
    & "Where Charindex( ?,  Title) > 0
Set qd = cn.CreateQuery("FindTitle", SQL)
qd(0).Type = rdTypeChar
```

Note You do not have to surround text parameters with quotes as this is handled automatically by the ODBC API interface.

Handling Output and Return Status Arguments

In some cases, a stored procedure returns an output or return status argument instead of or in addition to any rows returned by a SELECT statement. Each of these parameters must also be marked in the SQL statement with a question mark. Using this technique, you can mark the position of any number of parameters in your SQL query – including input, output or input/output.

Whenever your query returns output or return status arguments, you *must* use the ODBC CALL syntax when setting the SQL property of the **rdoQuery** object. In this case, a typical stored procedure call would look like this:

```
Dim qd as rdoQuery, rd as rdoResultset, SQL as String
SQL = "{ ? = Call master..sp_password (?, ?) }"
Set qd = CreateQuery ("SetPassword", SQL)
qd.rdoParameters(0).Direction = rdParamReturnValue
qd(1) = "Fred"    ' the old password
qd(2) = "George"  ' the new password
set rd = qd.Execute
if qd(0) <> 0 then _
   MsgBox "Operation failed"
```

Tip Be sure to specifically address stored procedures that do not reside in the current (default) database. In this example, the default database is *not* Master where the sp_password procedure is maintained, so this procedure is specifically addressed.

When control returns to your application after the procedure is executed, the **rdoParameter** objects designated as **rdParamReturnValue**, **rdParamOutput** or **rdParamInputOutput** contain the returned argument values. In the example shown above, the return status is available by examining `qd(0)` after the query is executed.

Using Other Properties

Using the properties of an **rdoParameter** object, you can set a query parameter that can be changed before the query is run. You can:

- Use the **Direction** property setting to determine if the parameter is an input, output, or input/output parameter, or a return value. In RDO 2.0, the **Direction** property is usually set automatically, so it is unnecessary to set this value. It is also unnecessary to set it for input parameters — which is the default value.
- Use the **Type** property setting to determine the data type of the **rdoParameter**. Data types are identical to those specified by the **rdoColumn.Type** property. In some cases, RDO might not be able to determine the correct parameter data type. In these cases, you can force a specific data type by setting the **Type** property.
- Use the **Value** property (the default property of an **rdoParameter**) to pass values to the SQL queries containing parameter markers used in **rdoQuery.Execute** or **rdoQuery.OpenResultset** methods. For example:

  ```
  MyQuery(0) = 5
  ```

Note RDO requires that your ODBC driver support a number of Level II compliant options and support the **SQLNumParams, SQLProcedureColumns** and **SQLDescribeParam** ODBC API functions in order to be able to create the **rdoParameters** collection and parse parameter markers in SQL statements. While some drivers can be used to create and execute queries, if your driver does not support creation of the **rdoParameters** collection, RDO fails quietly and simply does not create the collection. As a result, any reference to the collection results in a trappable error.

Addressing the Parameters

By default, members of the **rdoParameters** collection are named "Param*n*" where *n* is the **rdoParameter** object's ordinal number. For example, if an **rdoParameters** collection has two members, they are named "Param0" and "Param1". However, if you use the User Connection Designer, you can specify names for specific parameters.

Because the **rdoParameters** collection is the default collection for the **rdoQuery** object, addressing parameters is easy. Assuming you have created an **rdoQuery** object referenced by `rdoQo`, you can refer to the **Value** property of its **rdoParameter** objects by:

- Referencing the **Name** property setting using this syntax:

```
' Refers to PubDate parameter
rdoQo("PubDate")
```

-Or-

```
' Refers to PubDate parameter
rdoQo!PubDate
```

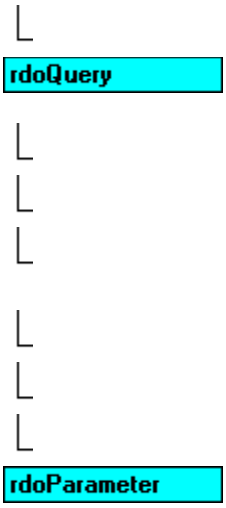
- Referencing its ordinal position in the **rdoParameters** collection using this syntax:

```
' Refers to the first parameter marker
rdoQo(0)
```

rdoParameters Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoParameterSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoParameterSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoParametersP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":""} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoParametersCollectionS"}

An **rdoParameters** collection contains all the **rdoParameter** objects of an **rdoQuery** object.



Remarks

The **rdoParameters** collection provides information only about *marked* parameters in an **rdoQuery** object or stored procedure. You can't append objects to or delete objects from the **rdoParameters** collection.

When the **rdoParameters** collection is first referenced, RDO and the ODBC interface parse the query searching for parameter markers – the question mark (?). For each marker found, RDO creates an **rdoParameter** object and places it in the **rdoParameters** collection. However, if the query cannot be compiled or otherwise processed, the **rdoParameters** collection is *not* created and your code will trigger a trappable error indicating that the object does not exist. In this case, check the query for improper syntax, permissions on underlying objects, and proper placement of parameter markers.

rdoParameter Object, rdoParameters Collection, Direction Property Example

This example executes a stored procedure against the SQL Server 'Pubs' database. The procedure text is also included here so you can setup this example on your own machine. The stored procedure expects your code to provide three input arguments: A string to use in an expression to choose the title, and two numbers used to choose a price range for the books. The procedure returns the number of books that fall in the range, and the maximum price of the books. It also returns a set of rows containing detailed information about the books.

To establish the connection, we assume the name of the server is "SEQUEL" and it is a Microsoft SQL Server – this is a DSN-less connection. Next, we use the ODBC CALL syntax to prepare the query. Notice that each parameter is marked with a question mark. Once, marked, the **rdoParameters** collection is used to set the direction for the output and return value parameters and the initial values for the input parameters. While you don't see the **rdoParameters** collection called out specifically, understand that it is the default collection of the **rdoQuery** object so references are made simpler by *not* including a reference to the **rdoParameters** collection itself.

```
Sub RunQuery_Click()
Dim rs As rdoResultset
Dim cn As New rdoConnection
Dim qd As New rdoQuery
Dim cl As rdoColumn
Const None As String = ""

cn.Connect = "uid=;pwd=;server=SEQUEL;" _
    & "driver={SQL Server};database=pubs;" _
    & "DSN='';"
cn.CursorDriver = rdUseOdbc
cn.EstablishConnection rdDriverNoPrompt

Set qd.ActiveConnection = cn
qd.SQL = "{ ? = Call ShowOutputRS (?,?,?,?,?) }"
qd(0).Direction = rdParamReturnValue
qd(4).Direction = rdParamOutput
qd(5).Direction = rdParamOutput
qd(1) = "c"
qd(2) = 5
qd(3) = 50

Set rs = qd.OpenResultset(rdOpenForwardOnly, _
    rdConcurReadOnly)

For Each cl In rs.rdoColumns
    Debug.Print cl.Name,
Next
Debug.Print

Do Until rs.EOF
    For Each cl In rs.rdoColumns
        Debug.Print cl.Value,
    Next
    rs.MoveNext
Debug.Print
Loop

Debug.Print "Output from SP="; qd(3)
Debug.Print "Return Status from SP="; qd(0)
```

```
rs.Close
qd.Close
cn.Close

End Sub
```

This is the stored procedure that is executed by the example shown above.

```
CREATE PROCEDURE ShowOutputRS
(
   @Ser varChar(128),
   @PriceLow Integer,
   @PriceHigh Integer,
   @Hits Integer OUTPUT,
   @MaxPrice integer OUTPUT
)
AS
Select @MaxPrice = Max(Price) from Titles
where Charindex(@Ser, title) > 0
and price between @priceLow and @priceHigh

Select * from Titles
where Charindex(@Ser, title) > 0
and price between @priceLow and @PriceHigh

Select @Hits = @@RowCount

return @@ROWCOUNT
```

rdoPreparedStatement Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoPreparedStatementC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoPreparedStatementX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoPreparedstatementP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoPreparedstatementM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoPreparedstatementS"}

An **rdoPreparedStatement** object is a prepared query definition.

Remarks

Note The **rdoPreparedStatement** object is outdated and only maintained for backward compatibility. It should be replaced with the **rdoQuery** object. The **rdoQuery** object supports all of the **rdoPreparedStatement** object's properties and methods. In contrast, the **rdoPreparedStatement** only a subset of the **rdoQuery** object's properties and methods and none of its events.

rdoPreparedStatements Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoPreparedstatementSC;rdobjrdoPreparedStatementsCollectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoPreparedStatementsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":""} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoPreparedStatementsCollectionS"}

An **rdoPreparedStatements** collection contains all **rdoPreparedStatement** objects in an **rdoConnection**.

Remarks

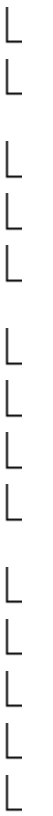
Note The **rdoPreparedStatements** collection is outdated and maintained for compatibility. It should be replaced with the **rdoQueries** collection. The **rdoQuery** object and **rdoQueries** collection supports all of the **rdoPreparedStatement** object's properties and methods. In contrast, the **rdoPreparedStatement** supports only a subset of the **rdoQuery** object's properties and methods and none of its events.

Note RDO requires that your ODBC driver support a number of Level II options and support the **SQLNumParams, SQLProcedureColumns** and **SQLDescribeParam** ODBC API functions in order to be able to create the **rdoParameters** collection and parse SQL statement parameter markers. While some drivers can be used to create and execute queries, if your driver does not support creation of the **rdoParameters** collection, RDO fails quietly and simply does not create the collection.

rdoQuery Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoQueryC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoQueryX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoQueryP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoQueryM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoQueryS"}

An **rdoQuery** object is a query definition that can include zero or more parameters.



Remarks

The **rdoQuery** object is used to manage SQL queries requiring the use of input, output or input/output parameters. Basically, an **rdoQuery** functions as a compiled SQL statement. When working with stored procedures or queries that require use of arguments that change from execution to execution, you can create an **rdoQuery** object to manage the query parameters. If your stored procedure returns output parameters or a return value, or you wish to use **rdoParameter** objects to handle the parameters, you *must* use an **rdoQuery** object to manage it. For example, if you submit a query that includes information provided by the user such as a date range or part number, RDO can substitute these values automatically into the SQL statement when the query is executed.

Note The **rdoQuery** object replaces the outdated **rdoPreparedStatement** object. The **rdoQuery** object remains similar to the **rdoPreparedStatement** in its interface, but adds the ability to be persisted into a Visual Basic project, allowing you to create and manipulate it at design time. Additionally, the **rdoQuery** objects can be prepared or not, allowing the you to choose the most appropriate use of the query.

Creating rdoQuery Objects

To create an **rdoQuery** object, use the **CreateQuery** method which associates the **rdoQuery** with a specific **rdoConnection** object and adds it to the **rdoQueries** collection. Once created, you must fill

in required parameters using the **rdoParameters** collection, and then use the **OpenResultset** method to create resultsets from the query, or the **Execute** method to simply run the query if it does not return rows.

You can also use the User Connection Designer (CQD) to create **rdoQuery** objects in your project. The CQD takes your SQL query and permits you to specify the data types for each parameter. It then inserts appropriate code in your application to expose these procedures very much like methods off of the **rdoQuery** object.

Note Due to the extra overhead involved in creating and managing **rdoQuery** objects and the **rdoParameters** collection, you should not use parameter queries for SQL statements that do not change from execution to execution — especially those that are executed only once or infrequently.

Stand Alone rdoQuery Objects

You can declare a stand-alone **rdoQuery** object using the **Dim** statement as follows:

```
Dim MyQuery as New rdoQuery
```

Stand-alone **rdoQuery** objects are not assigned to a specific **rdoConnection** object, so you must set the **ActiveConnection** property before attempting to execute the query, or to use the **OpenResultset** object against it. The **CursorType** and **ErrorThreshold** properties are set from default values established by the **rdoEngine** default settings. In addition, new **rdoQuery** objects are not automatically appended to the **rdoQueries** collection until you use the **Add** method.

For example, the code shown below creates an **rdoQuery** object, associates it with a connection, and executes it. Next, the **rdoQuery** object is associated with a different connection and executed again. The query object becomes more of an encapsulation of any kind of query, and thus can be executed against any kind of connection, provided the SQL statement would be appropriate for the connection.

```
Dim MyQuery As rdoQuery '
MyQuery.SQL = "Update customers " _
   & " Set LastTouched = GetDate()"
MyQuery.Prepared = False   'don't prepare it,
                           'just SQLExecDirect
'assume that cnSomeConnection
'is an rdoConnection or stand-alone object
MyQuery.ActiveConnection = cnSomeConnection
MyQuery.Execute

MyQuery.ActiveConnection = cnOtherConnection
'the cnOtherConnection is over a WAN, so I can increase
'my query timeout to compensate
MYQuery.QueryTimeout = 120
MyQuery.Execute
```

Choosing the right SQL Syntax

When coding the SQL property of an **rdoQuery** object, you can choose between one of three syntax styles to code your parameter query:

- εδ Στρινγσ: Your code builds up the SQL statement and its parameters using the Visual Basic concatenation (&) operator. This statement can be passed to the ***SQL*** argument of the **OpenResultset** method or the **rdoQuery** object's **SQL** property. In this case, a parameter query might look like this:

```
sSQL = "Select Name, Age From Animals " _
   & " Where Weight > " & WeightWanted.Text _
   & " and Type = ' & TypeWanted.Text & "'"
```

- ΣΘΛ σψνταξ: The SQL syntax used by the remote server. In this case you can execute your own query or stored procedure, and pass in parameters by concatenation, or using placeholders, or

both. The parameters marked with placeholders are managed by RDO as **rdoParameter** objects. A parameter query might look like this:

```
sSQL = "Select Au_LName from Authors" _
   & " Where Au_Fname = ?"
```

– Or –

```
sSQL = "Execute MyStoredProc 'Arg1', 450, '"  _
   & Text1
```

– Or –

```
sSQL = "Execute MyStoredProc ?, ?, ?"
```

•ΑΛΛ σψνταξ: Designed to call stored procedures that return a return status or output parameters. In this case, a placeholder can be defined for each input, output, or input/output parameter which is automatically mapped to **rdoParameter** objects. You can also mix in concatenated operators as needed. In this case, a parameter query might look like this:

```
sSQL = "{call ParameterTest (?,?,?) }"
```

– Or –

```
sSQL = "{? = call ParameterTest (?,?,?) }"
```

– Or –

```
sSQL = "{? = call CountAnimals (?, ?, 14, 'Pig')}
```

The **rdoQuery** object is managed by setting the following properties and methods.

- Use the **SQL** property to specify a parameterized SQL statement to execute. The ***name*** argument of the **CreateQuery** method can also be used to provide the SQL query string.
- Set query parameters by using the **rdoQuery** object's **rdoParameters** collection.
- Use the **Prepared** property to indicate if the **rdoQuery** object should be prepared by the ODBC **SQLPrepare** function. If **False**, the query is executed using the **SQLExecDirect** function.
- Use the **Type** property to determine whether the query selects rows from an existing table (select query), performs an action (an action query), contains both action and select operations, or represents a stored procedure.
- Use the **RowsetSize** property setting to determine how many rows are buffered internally when building a cursor and locked when using pessimistic locking.
- Use the **KeysetSize** property to indicate the size of the keyset buffer when creating cursors.
- Use the **MaxRows** property to indicate the maximum number of rows to be returned by a query.
- Use the **RowsAffected** property to indicate how many rows are affected by an action query.
- Use the **QueryTimeout** property to indicate how long the driver manager waits before pausing a query and firing the QueryTimeout event.
- Use the **BindThreshold** property to indicate the largest column to be automatically bound.
- Use the **ErrorThreshold** property to indicate the error level that constitutes a trappable error.
- Use the **Updatable** property to see if the result set generated by an **rdoQuery** can be updated.
- Use the **OpenResultset** method to create an **rdoResultset** based on the **OpenResultset** arguments and properties of the **rdoQuery**.
- Use the **Execute** method to run an action query using **SQL** and other **rdoQuery** properties, including any values specified in the **rdoParameters** collection.
- Use the **LogMessages** property to activate ODBC tracing.

rdoQuery Object Events

The following events are fired as the **rdoQuery** object is manipulated. These can be used to micro-manage queries associated with the **rdoQuery** or coordinate other processes in your application.

Event Name	Description
QueryComplete	Fired when a query has completed.

QueryTimeout	Fired when the **QueryTimeout** period has elapsed and the query has not begun to return rows.
WillExecute	Fired before the query is executed permitting last-minute changes to the SQL, or to prevent the query from executing.

Closing the rdoQuery Object

Use the **Close** method to close an **rdoQuery** object, set its **ActiveConnection** property to **Nothing**, and remove it from the **rdoQueries** collection. However, you can still re-associate the **rdoQuery** object with another **rdoConnection** object by setting its **ActiveConnection** property to another **rdoConnection** object. Using the **Execute** method or **OpenResultset** method against an **rdoQuery** object that has its **ActiveConnection** property set to **Nothing** or an invalid **rdoConnection** causes a trappable error.

Addressing rdoQuery Objects

rdoQuery objects are the preferred way to submit parameter queries to the external server. For example, you can create a parameterized Transact SQL query (as used on Microsoft SQL Server) and store it in an **rdoQuery** object.

You refer to an **rdoQuery** object by its **Name** property setting using the following syntax. Since the **rdoQuery** object's default collection is the **rdoParameters** collection, all unqualified references to the **rdoQuery** object refer to the **rdoParameters** collection. In these examples, assume we have created an **rdoQuery** object named `rdoQo`. The first two examples refer to the **rdoQuery** object named "MyQuery".

rdoQo("MyQuery")

– Or –

rdoQo!*MyQuery*

You can also refer to **rdoQuery** objects (and the **rdoPreparedStatements** collection) by their position in the **rdoQueries** collection using this syntax (where *n* is the *n*th member of the zero-based **rdoQueries** collection):

rdoQo(*n*)

rdoQueries Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoQueriesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoQueriesX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoQueriesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":""} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoQueriesCollectionS"}

Contains **rdoQuery** objects that have been added to the **rdoQueries** collection either automatically via the **CreateQuery** method, or with the **Add** method.

Remarks

An **rdoQuery** object is automatically appended to the **rdoQueries** collection when you use the **CreateQuery** method of the **rdoConnection** object. You can also use the **Add** method against the rdoQueries collection supplying a stand-alone **rdoQuery** object as the argument.

When you use the **Close** method against and **rdoQuery** object, it is removed from the **rdoQueries** collection, but the object remains instantiated. By resetting the **ActiveConnection** property, you can associate the **rdoQuery** object with another connection and use the **Add** method to append it to the **rdoQueries** collection.

An **rdoQuery** object need not be a member of the **rdoQueries** collection before it can be associated with an **rdoConnection** object and used with the **Execute** or **OpenResultset** methods.

rdoQuery Object, rdoQueries Collection Example

This example leverages RDO's ability to set the data type of individual arguments of a query. In this case, a CHARINDEX function argument is passed as a parameter. Since the ODBC driver does not recognize this data type correctly, we simply change it to CHAR before assigning a value to the parameter. The query itself uses TSQL syntax – it does not need to use the ODBC CALL syntax as it does not execute a parameter-based stored procedure. This example also creates a DSN-less connection to a Microsoft SQL Server and uses the sample Pubs database.

```
Private Sub Query1_Click()
Dim rs As rdoResultset
Dim cn As New rdoConnection
Dim qd As New rdoQuery
Dim cl As rdoColumn
Const None As String = ""

cn.Connect = "uid=;pwd=;server=SEQUEL;" _
    & "driver={SQL Server};database=pubs;" _
    & "DSN='';"
cn.CursorDriver = rdUseOdbc
cn.EstablishConnection rdDriverNoPrompt

Set qd.ActiveConnection = cn
qd.SQL = "Select * From Titles" _
    & " Where CharIndex( ?, Title) > 0"

qd(0).Type = rdTypeCHAR
qd(0) = InputBox("Enter search string", , "C")

Set rs = qd.OpenResultset(rdOpenForwardOnly, rdConcurReadOnly)

For Each cl In rs.rdoColumns
    Debug.Print cl.Name,
Next
Debug.Print

Do Until rs.EOF
    For Each cl In rs.rdoColumns
        Debug.Print cl.Value,
    Next
    rs.MoveNext
Debug.Print
Loop
End Sub
```

rdoResultset Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoResultsetC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoResultSetX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoResultsetP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoResultsetM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoResultSetE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoResultSetS"}

An **rdoResultset** object represents the rows that result from running a query

Remarks

When you use remote data objects, you interact with data almost entirely using **rdoResultset** objects. **rdoResultset** objects are created using the RemoteData control, or the **OpenResultset** method of the **rdoQuery**, **rdoTable,** or **rdoConnection** object.

When you execute a query that contains one or more SQL SELECT statements, the data source returns zero or more rows in an **rdoResultset** object. All **rdoResultset** objects are constructed using rows and columns.

A single **rdoResultset** can contain zero or any number of result sets — so-called "multiple" result sets. Once you have completed processing the first result set in an **rdoResultset** object, use the **MoreResults** method to discard the current **rdoResultset** rows and activate the next **rdoResultset**. You can process individual rows of the new result set just as you processed the first **rdoResultset**. You can repeat this until the **MoreResults** method returns **False**.

A new **rdoResultset** is automatically added to the **rdoResultsets** collection when you open the object, and it's automatically removed when you close it.

Note RDO 1.0 collections behave differently than Data Access Object (DAO) collections. When you **Set** a variable containing a reference to a RDO object like **rdoResultset**, the existing **rdoResultset** is *not* closed and removed from the **rdoResultsets** collection. The existing object remains open and a

member of its respective collection.

In contrast, RDO 2.0 collections do not behave in this manner. When you use the **Set** statement to assign a variable containing a reference to an RDO object, the existing object *is* closed and removed from the associated collection. This change is designed to make RDO more compatible with DAO.

Processing Multiple Result Sets

When you execute a query that contains more than one SELECT statement, you must use the **MoreResults** method to discard the current **rdoResultset** rows and activate each subsequent **rdoResultset**. Each of the **rdoResultset** rows *must* be processed or discarded before you can process subsequent result sets. To process result set rows, use the *Move* methods to position to individual rows, or the **MoveLast** method to position to the last row of the **rdoResultset**. You can use the **Cancel** or **Close** methods against **rdoResultset** objects that have not been fully processed.

Choosing a Cursor Type

You can choose the type of **rdoResultset** object you want to create using the *type* argument of the **OpenResultset** method – the default **Type** is **rdOpenForwardOnly** for RDO and **rdOpenKeyset** for the **RemoteData** control. If you specify **rdUseNone** as the **CursorDriver** property, a forward-only, read-only result set is created. Each type of **rdoResultset** can contain columns from one or more tables in a database.

There are four types of **rdoResultset** objects based on the type of cursor that is created to access the data:

- Forward-only — type **rdoResultset** — individual rows in the result set can be accessed and can be updatable (when using server-side cursors), but the current row pointer can only be moved toward the end of the **rdoResultset** using the **MoveNext** method — no other method is supported.
- Static-type **rdoResultset** — a static copy of a set of rows that you can use to find data or generate reports. Static cursors might be updatable when using either the ODBC cursor library or server-side cursors, depending on which drivers are supported and whether the source data can be updated.
- Keyset-type **rdoResultset** — the result of a query that can have updatable rows. Movement within the keyset is unrestricted. A keyset-type **rdoResultset** is a dynamic set of rows that you can use to add, change, or delete rows from an underlying database table or tables. Membership of a keyset **rdoResultset** is fixed.
- Dynamic-type **rdoResultset** — the result of a query that can have updatable rows. A dynamic-type **rdoResultset** is a dynamic set of rows that you can use to add, change, or delete rows from an underlying database table or tables. Membership of a dynamic-type **rdoResultset** is not fixed.

Dissociate rdoResultset objects

When using the client batch cursor library, RDO permits you to disconnect an **rdoResultset** object from the **rdoConnection** object used to populate its rows by setting the **ActiveConnection** property to **Nothing**. While dissociated, the **rdoResultset** object becomes a temporary static snapshot of a local cursor. It can be updated, new rows can be added and rows can be removed from this **rdoResultset**. You can re-associate the **rdoResultset** by setting the **ActiveConnection** property to another (or the same) **rdoConnection** object. Once reconnected, you can use the **BatchUpdate** method to synchronize the **rdoResultset** with a remote database.

To perform this type of dissociated update operation, you should open the **rdoResultset** using an **rdOpenStatic** cursor, and use the **rdConcurBatch** as the concurrency option.

Managing rdoResultset Object Properties and Methods

You can use the methods and properties of the **rdoResultset** object to manipulate data and navigate the rows of a result set. For example, you can:

- Use the **Type** property to indicate the type of **rdoResultset** created, and the **Updatable** property

indicates whether or not you can change the object's rows.

- Use the **BOF** and **EOF** properties to see if the current row pointer is positioned beyond either end of the **rdoResultset** or it contains no rows.
- Use the **MoveNext** method to reposition the current row in forward-only type **rdoResultset** objects.
- Use the **Bookmarkable, Transactions**, and **Restartable** properties to determine if the **rdoResultset** supports bookmarks or transactions, or can be restarted.
- Use the **LockEdits** property to determine the type of locking used to update the **rdoResultset**.
- Use the **RowCount** property to determine how many rows in the **rdoResultset** are available. If the **RowCount** property returns -1, RDO cannot determine how many rows have been processed. Only when you move to **EOF** does the **RowCount** property reflect the number of rows returned by the query. Not all cursor types support this functionality. The **RowCount** property returns -1 if it is not available.
- Use the **AddNew**, **Edit**, **Update**, and **Delete** methods to add new rows or otherwise modify updatable **rdoResultset** objects. Use the **CancelUpdate** method to cancel pending edits.
- Use the **Requery** method to restart the query used to create an **rdoResultset** object. This method can be used to re-execute a parameterized query.
- Use the **MoreResults** method to complete processing of the current **rdoResultset** and begin processing the next result set generated from a query. Use the **Cancel** method to terminate processing of all pending queries when the query contains more than one SQL operation. When you use the **Close** method against an **rdoResultset**, all pending queries are flushed and the **rdoResultset** is automatically dropped from the **rdoResultsets** collection.
- Use the **Close** method to terminate and deallocate the **rdoResultset** object and remove it from the **rdoResultsets** collection.

rdoResultset Events

The following events are fired as the **rdoResultset** object is manipulated. These can be used to micro-manage result sets or to synchronize other processes with the operations performed on the **rdoResultset** object.

Event Name	Description
Associate	Fired after a new connection is associated with the object.
ResultsChange	Fired after current rowset is changed (multiple result sets).
Dissociate	Fired after the connection is set to nothing.
QueryComplete	Fired after a query has completed.
RowStatusChange	Fired after the state of the current row has changed (edit, delete, insert).
RowCurrencyChange	Fired after the current row pointer is repositioned.
WillAssociate	Fired before a new connection is associated with the object.
WillDissociate	Fired before the connection is set to nothing.
WillUpdateRows	Fired before an update to the server occurs.

Executing Multiple Operations on a Connection

If there is an unpopulated **rdoResultset** pending on a data source that can only support a single operation on an **rdoConnection** object, you cannot create additional **rdoQuery** or **rdoResultset** objects**,** or use the **Refresh** method on the **rdoTable** object until the **rdoResultset** is flushed, closed, or fully populated. For example, when using SQL Server 4.2 as a data source, you cannot create an additional **rdoResultset** object until you move to the last row of the last result set of the current

rdoResultset object. To populate the result set, use the **MoreResults** method to move through all pending result sets, or use the **Cancel** or **Close** method on the **rdoResultset** to flush all pending result sets.

Handing Beginning and End of File Conditions

When you create an **rdoResultset**, the current row is positioned to the first row if there are any rows. If there are no rows, the **RowCount** property setting is 0, and the **BOF** and **EOF** property settings are both **True**.

Note An **rdoResultset** may not be updatable even if you request an updatable **rdoResultset**. If the underlying database, table, or column isn't updatable, or if your user does not have update permission, all or portions of your **rdoResultset** may be read-only. Examine the **rdoConnection**, **rdoResultset,** and **rdoColumn** objects' **Updatable** property to determine if your code can change the rows.

Closing rdoResultset objects

Use the **Close** method to remove an **rdoResultset** object from the **rdoResultsets** collection, disassociate it from its connection, and free all associated resources. No events are fired when you use the **Close** method.

Setting the **ActiveConnection** property to **Nothing** removes the **rdoResultset** object from the **rdoResultsets** collection and fires events, but does not deallocate the object resources. Setting the **rdoResultset** object's **ActiveConnection** property to a valid **rdoConnection** object causes the **rdoResultset** object to be re-appended to the **rdoResultsets** collection of the **rdoConnection** object.

Addressing rdoResultset Objects

The default collection of an **rdoResultset** is the **rdoColumns** collection, and the default property of an **rdoColumn** object is the **Value** property. You can simplify your code by taking advantage of these defaults. For example, the following lines of code all set the value of the `PubID` column in the current row of an **rdoResultset**:

```
MyRs.rdoColumns("PubID").Value = 99
MyRs("PubID") = 99
MyRs!PubID = 99
' This is the first column
' returned by the SELECT statement...
MyRs(0) = 99
```

The **Name** property of an **rdoResultset** object contains the first 255 characters of the query used to create the resultset, so it is often unsuitable as an index into the **rdoResultsets** collection especially since several queries might be created with the same SQL query.

You can refer to **rdoResultset** objects by their position in the **rdoResultsets** collection using this syntax (where *n* is the *n*th member of the zero-based **rdoResultsets** collection):

```
rdoResultsets(n)
```

rdoResultsets Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoResultsetSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoResultsetSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoResultsetsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":""} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoResultsetsCollectionS"}

The **rdoResultsets** collection contains all open **rdoResultset** objects in an **rdoConnection**.

Remarks

A new **rdoResultset** is automatically added to the **rdoResultsets** collection when you open the object, and it's automatically removed when you close it. Several **rdoResultset** objects might be active at any one time.

Use the **Close** method to remove an **rdoResultset** object from the **rdoResultsets** collection, disassociate it from its connection, and free all associated resources. No events are fired when you use the **Close** method.

Setting the **ActiveConnection** property to **Nothing** removes the **rdoResultset** object from the **rdoResultsets** collection and fires events, but does not deallocate the object resources. Setting the **rdoResultset** object's **ActiveConnection** property to a valid **rdoConnection** object causes the **rdoResultset** object to be re-appended to the **rdoResultsets** collection.

Note RDO 1.0 collections behave differently than Data Access Object (DAO) collections. When you **Set** a variable containing a reference to a RDO object like **rdoResultset**, the existing **rdoResultset** is *not* closed and removed from the **rdoResultsets** collection. The existing object remains open and a member of its respective collection.

In contrast, RDO 2.0 collections do not behave in this manner. When you use the Set statement to assign a variable containing a reference to an RDO object, the existing object *is* closed and removed from the associated collection. This change is designed to make RDO more compatible with DAO.

Managing the rdoResultsets Collection

When you use the **OpenResultset** method against an **rdoConnection** or **rdoQuery**, and assign the result to an existing **rdoResultset** object, the existing object is maintained and a new **rdoResultset** object is appended to the **rdoResultsets** collection. When performing similar operations using the Microsoft Jet database engine and Data Access Objects (DAO), existing recordset objects are automatically closed when the variable is assigned, and no two **Recordsets** collection members can have the same name. For example, using RDO:

```
Dim rs as rdoResultset
Dim cn as rdoConnection
Set cn = OpenConnection....
Set rs = cn.OpenResultset("Select * from Authors", _
    rdOpenStatic)
Set rs = cn.OpenResultset("Select * from Titles", _
    rdOpenDynamic)
```

This code opens two separate **rdoResultset** objects; both are stored in the **rdoResultsets** collection. After this code runs, the second query, which is stored in **rdoResultsets**(1), is assigned to the **rdoResultset** variable `rs`. The first query is available and its cursor is still available by referencing **rdoResultsets**(0). Because of this implementation, more than one member of the **rdoResultsets** collection can have the same name.

This behavior permits you to maintain existing **rdoResultset** objects, which are maintained in the **rdoResultsets** collection, or close them as needed. In other words, you must explicitly close any **rdoResultset** objects that are no longer needed. Simply assigning another **rdoResultset** to a **rdoResultset**-type variable has no affect on the existing **rdoResultset** formerly referenced by the variable. Note that the procedures and other temporary objects created to manage the **rdoResultset** are maintained on the remote server as long as the **rdoResultset** remains open.

If you write an application that does not close each **rdoResultset** before opening additional **rdoResultset** objects, the number of procedures maintained in *TempDB* or elsewhere on the server increases each time another **rdoResultset** object is opened. In addition those resultsets might require significant client or server resources to store keysets or row values. Over time, this behavior can overflow the capacity of the server or workstation resources.

rdoResultset Object, rdoResultsets Collection Example

The following example illustrates execution of a multiple result set query. While this query uses three SELECT statements, only two return rows to your application. The subquery used instead of a join does not pass rows outside the scope of the query itself. This is also an example of a simple parameter query that concatenates the arguments instead of using an **rdoQuery** to manage the query. The **OpenResultset** also runs asynchronously – the code checks for completion of the operation by polling the **StillExecuting** property.

```
Private Sub ShowResultset_Click()
Dim rs As rdoResultset
Dim cn As New rdoConnection
Dim cl As rdoColumn
Dim SQL As String
Const None As String = ""

cn.Connect = "uid=;pwd=;server=SEQUEL;" _
    & "driver={SQL Server};database=pubs;" _
    & "DSN='';"

cn.CursorDriver = rdUseOdbc
cn.EstablishConnection rdDriverNoPrompt

SQL = "Select Au_Lname, Au_Fname" _
    & " From Authors A" _
    & " Where Au_ID in " _
    & " (Select Au_ID" _
    & "     from TitleAuthor TA, Titles T" _
    & "     Where TA.Au_ID = A.Au_ID" _
    & "     And TA.Title_ID = T.Title_ID " _
    & "     And T.Title Like '" _
    & InputBox("Enter search string", , "C") & "%')" _
    & "Select * From Titles Where price > 10"

Set rs = cn.OpenResultset(SQL, rdOpenKeyset, _
    rdConcurReadOnly, rdAsyncEnable + rdExecDirect)

Debug.Print "Executing ";
While rs.StillExecuting
    Debug.Print ".";
    DoEvents
Wend

Do
    Debug.Print String(50, "-") _
   & "Processing Result Set " & String(50, "-")
    For Each cl In rs.rdoColumns
        Debug.Print cl.Name,
    Next
    Debug.Print

    Do Until rs.EOF
        For Each cl In rs.rdoColumns
            Debug.Print cl.Value,
        Next
        rs.MoveNext
```

```
    Debug.Print
    Loop
    Debug.Print "Row count="; rs.RowCount

Loop Until rs.MoreResults = False
End Sub
```

rdoTable Object

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoTableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoTableX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoTableP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoTableM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoTableS"}

An **rdoTable** object represents the stored definition of a base table or an SQL view.

Remarks

Note You are discouraged from using the **rdoTable** object and **rdoTables** collection to manage or inspect the structure of your database tables. This object is maintained for backward compatibility and might not be supported in future versions of Visual Basic or RDO.

You can map a table definition using an **rdoTable** object and determine the characteristics of an **rdoTable** object by using its methods and properties. For example, you can:

- Examine the column properties of any table in an ODBC database. (Note that all **rdoTable** object properties are read-only.)
- Use the **OpenResultset** method to create an **rdoResultset** object based on all of the rows of the base table.
- Use the **Name** property to determine the name of the table or view.
- Use the **RowCount** property to determine the number of rows in the table or view. Referencing the **RowCount** property causes the query to be completed — just as if you had used the **MoveLast** method.
- Use the **Type** property to determine the type of table. The ODBC data source driver determines the supported table types.
- Use the **Updatable** property to determine if the table supports changes to its data.

You cannot reference the **rdoTable** objects until you have populated the **rdoTables** collection because it is not automatically populated when you connect to a data source. To populate the **rdoTables** collection, use the **Refresh** method or reference individual members of the collection by their ordinal number.

When you use the **OpenResultset** method against an **rdoTable** object, RDO executes a "SELECT * FROM *table*" query that returns *all* rows of the table using the cursor type specified. By default, a forward-only cursor is created.

You cannot define new tables or change the structure of existing tables using RDO or the RemoteData control. To change the structure of a database or perform other administrative functions, use SQL queries or the administrative tools that are provided with the database.

The default collection of an **rdoTable** object is the **rdoColumns** collection. The default property of an **rdoTable** is the **Name** property. You can simplify your code by using these defaults. For example, the following statements are identical in that they both print the number corresponding to the column data type of a column in an **rdoTable** using a RemoteData control:

```
Print RemoteData1.Connection.rdoTables _
   ("Publishers").rdoColumns("PubID").Type
Print RemoteData1.Connection("Publishers"). _
   ("PubID").Type
```

The **Name** property of an **rdoTable** object isn't the same as the name of an object variable to which it's assigned — it is derived from the name of the base table in the database.

You refer to an **rdoTable** object by its **Name** property setting using this syntax:

```
rdoTables("Authors") 'Refers to the Authors table
```

– Or –

```
rdoTables!Authors    'Refers to the Authors table
```

You can also refer to **rdoTable** objects by their position in the **rdoTables** collection using this syntax (where *n* is the *n*th member of the zero-based **rdoTables** collection):

```
rdoTables(n)
```

rdoTables Collection

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoTableSC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoTableSX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdidxrdoTablesP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdidxrdoTablesM"} {ewc HLP95EN.DLL,DYNALINK,"Events":""} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoTablesCollectionS"}

The **rdoTables** collection contains all stored **rdoTable** objects in a database.

Remarks

Note You are discouraged from using the **rdoTable** object and **rdoTables** collection to manage or inspect the structure of your database tables. This object is maintained for backward compatibility and might not be supported in future versions of Visual Basic.

For performance reasons, you cannot reference an **rdoTable** object until you have first populated the **rdoTables** collection because it is not automatically populated when you connect to a data source. To populate the **rdoTables** collection, use the **Refresh** method or reference individual members of the collection by their ordinal number. Depending on the number of tables in your database, this can take quite some time.

Remote Data Objects and Collections

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjrdoObjectsCollectionsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjrdoObjectsCollectionsX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdobjrdoObjectsCollectionsP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdobjrdoObjectsCollectionsM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjrdoObjectsCollectionsE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjrdoObjectsCollectionsS"}

Remote Data objects and collections provide a framework for using code to create and manipulate components of a remote ODBC database system. Objects and collections have properties that describe the characteristics of database components and methods that you use to manipulate them. Using the containment framework, you create relationships among objects and collections, and these relationships represent the logical structure of your database system.

Objects and collections provide different types of containment relationships: Objects contain zero or more collections, all of different types; and collections contain zero or more objects, all of the same type. Although objects and collections are similar entities, the distinction differentiates the two types of relationships.

Note The RDO is only supported on 32-bit operating systems such as Windows 95 and Windows NT. To use the Remote Data Objects, you must set a reference to the Microsoft Remote Data Object 2.0 object library in the Visual Basic References dialog box.

In the following table, the type of collection in the first column contains the type of object in the second column. The third column describes what each type of object represents.

Collection	Object	Description
rdoConnections	**rdoConnection**	An open or allocated connection
None	**rdoEngine**	The remote database engine
rdoErrors	**rdoError**	Information about ODBC errors
rdoEnvironments	**rdoEnvironment**	A logical set of **rdoConnection** objects with a common user name and password
rdoColumns	**rdoColumn**	A column that is part of an **rdoResultset**
rdoParameters	**rdoParameter**	A parameter for an

		rdoQuery or an **rdoPreparedStatement**
rdoPreparedStatements	**rdoPreparedStatements**	A saved query definition (outdated)
rdoQueries	**rdoQuery**	A saved query definition
rdoResultsets	**rdoResultset**	The rows resulting from a query
rdoTables	**rdoTable**	A table definition

RemoteData Control

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdobjRemoteDataC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdobjRemoteDataX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"rdobjRemoteDataP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"rdobjRemoteDataM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"rdobjRemoteDataE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdobjRemoteDataS"}

Provides access to data stored in a remote ODBC data source through bound controls. The **RemoteData** control enables you to move from row to row in a result set and to display and manipulate data from the rows in bound controls.

Syntax

RemoteData

Remarks

The **RemoteData** control provides an interface between Remote Data Objects (RDO) and data-aware bound controls. With the **RemoteData** control, you can:

- Establish a connection to a data source based on its properties.
- Create an **rdoResultset**.
- Pass the current row's data to corresponding bound controls.
- Permit the user to position the current row pointer.
- Pass any changes made to the bound controls back to the data source.

Overview

Without a **RemoteData** control, a **Data** control or its equivalent, data-aware (bound) controls on a form can't automatically access data. The RemoteData and Data controls are examples of *DataSource Controls*. You can perform most remote data access operations using the DataSource controls without writing any code at all. Data-aware controls bound to a DataSource control automatically display data from one or more columns for the current row or, in some cases, for a set of rows on either side of the current row. DataSource controls perform all operations on the current row.

The RemoteData DataSource Control

If the **RemoteData** control is instructed to move to a different row, all bound controls automatically pass any changes to the **RemoteData** control to be saved to the ODBC data source. The **RemoteData** control then moves to the requested row and passes back data from the current row to the bound controls where it's displayed.

The **RemoteData** control automatically handles a number of contingencies including empty result sets, adding new rows, editing and updating existing rows, converting and displaying complex data types, and handling some types of errors. However, in more sophisticated applications, you must trap some error conditions that the **RemoteData** control can't handle. For example, if the remote server has a problem accessing the data source, the user doesn't have permission, or the query can't be executed as coded, a trappable error results. If the error occurs *before* your application procedures start, or as a result of some internal errors, the Error event is triggered.

Operation

Use the **RemoteData** control properties to describe the data source, establish a connection, and specify the type of cursor to create. If you alter these properties once the result set is created, use the **Refresh** method to rebuild the underlying **rdoResultset** based on the new property settings.

The **RemoteData** control behaves like the Jet-driven **Data** control in most respects. The following guidelines illustrate a few differences that apply when setting the **SQL** property.

You can treat the **RemoteData** control's **SQL** property like the **Data** control's **RecordSource** property except that it cannot accept the name of a table by itself, unless you populate the **rdoTables** collection first. Generally, the **SQL** property specifies an SQL query. For example, instead of just "Authors", you would code "SELECT * FROM AUTHORS" which provides the same functionality. However, specifying a table in this manner is not a good programming practice as it tends to return too many rows and can easily exhaust workstation resources or lock large segments of the database.

The result set created by the **RemoteData** control might not be in the same order as the **Recordset** created by the **Data** control. For example, if the **Data** control's **RecordSource** property is set to "Authors" and the **RemoteData** control's **SQL** property is set to "SELECT * FROM AUTHORS", the first record returned by Jet to the **Data** control is based on the first available index on the Authors table. The **RemoteData** control, however, returns the first row returned by the remote database engine based on the physical sequence of the rows in the database, regardless of any indexes. In some cases, the order of the records could be identical, but not always.

This difference in behavior can affect how bound controls handle the resulting rows — especially multiple-row bound controls like the **DBGrid** control. You can manipulate the **RemoteData** control with the mouse — to move the current row pointer from row to row, or to the beginning or end of the **rdoResultset** by clicking the control. As you manipulate the **RemoteData** control buttons, the current row pointer is repositioned in the **rdoResultset**. You cannot move off either end of the **rdoResultset** using the mouse. You also can't set focus to the **RemoteData** control.

Other Features

You can use the objects created by the **RemoteData** control to create additional **rdoConnection, rdoResultset**, or **rdoQuery**objects.

You can set the **RemoteData** control **Resultset** property to an **rdoResultset** created independently of the control. If this is done, the **RemoteData** control properties are reset based on the new **rdoResultset** and **rdoConnection**.

You can set the **Options** property to enable asynchronous creation of the **rdoResultset** (**rdAsyncEnable**) or to execute the query without creating a temporary stored procedure (**rdExecDirect**).

The Validate event is triggered before each reposition of the current row pointer. You can choose to accept the changes made to bound controls or cancel the operation using the Validate event's *action* argument.

The **RemoteData** control can also manage what happens when you encounter an **rdoResultset** with no rows. By changing the **EOFAction** property, you can program the **RemoteData** control to enter **AddNew** mode automatically.

Programmatic Operation

To create an **rdoResultset** programmatically with the **RemoteData** control:

- Set the **RemoteData** control properties to describe the desired characteristics of the **rdoResultset**.
- Use the **Refresh** method to begin the automated process or to create the new **rdoResultset**. Any existing **rdoResultset** is discarded.

All of the **RemoteData** control properties and the new **rdoResultset** object may be manipulated independently of the **RemoteData** control—with or without bound controls. The **rdoConnection** and **rdoResultset** objects each have properties and methods of their own that can be used with procedures that you write.

For example, the **MoveNext** method of an **rdoResultset** object moves the current row to the next row in the **rdoResultset**. To invoke this method with an **rdoResultset** created by a **RemoteData** control, you could use this code:

```
RemoteData1.Resultset.MoveNext
```

AbsolutePosition Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproAbsolutePositionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproAbsolutePositionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproAbsolutePositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproAbsolutePositionS"}

Returns or sets the absolute row number of an **rdoResultset** object's current row.

Syntax

object.**AbsolutePosition** [= *value*]

The **AbsolutePosition** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Long Data Type value from -1 to the maximum number of rows in the **rdoResultset**. Corresponds to the ordinal position of the current row in the **rdoResultset** specified by *object*. Default value is -1.

Remarks

Use the **AbsolutePosition** property to position the current row pointer to a specific row based on its ordinal position in a keyset-, or static-type **rdoResultset**. It is *not* supported for dynamic or forward-only-type **rdoResultset** objects. While a value is returned for a dynamic cursor, the value is not necessarily accurate. Generally, the **rdoResultset** object's **Bookmarkable** property must be **True** before **AbsolutePosition** values are supported.

You can also determine the current row number by checking the **AbsolutePosition** property setting. For example, if you have populated 10 rows of a 50 row **rdoResultset**, the **AbsolutePosition** property returns 10. After you execute a **MoveLast** method against the result set, **AbsolutePosition** returns 50. You can then set the **AbsolutePosition** property to any value between 1 and 50 to position the current row pointer to that row.

The **AbsolutePosition** property value is -1 based thus a setting of 1 refers to the first row in the **rdoResultset**. Setting a value greater than the number of *populated* rows causes RDO to position to the last row in the result set (EOF).

If there is no current row, as when there are no rows in the **rdoResultset**, -1 is returned. If the current row is deleted, the **AbsolutePosition** property value isn't defined and a trappable error occurs if it is referenced. New rows are added to the end of the sequence if the type of cursor includes dynamic membership.

Note This property isn't intended to be used as a surrogate row number. Using bookmarks is still the recommended way of retaining and returning to a given position in a cursor. Also, there is no assurance that a given row will have the same absolute position if the **rdoResultset** is re-created because the order and membership of individual rows within an **rdoResultset** can vary between executions.

AbsolutePosition Property Example

The following example uses the SQL Server Pubs database to illustrate use of the **AbsolutePosition** property as a secondary letter index to a set of rows. The program begins by fetching the name and ID of all publishers into a dropdown **ComboBox** control. Initially, and as a specific publisher is chosen from the **ComboBox**, the set of titles for this publisher is fetched from the Titles table. This is accomplished by creating a query using the concatenation method and setting the **RemoteData** control's SQL property to this query. A **DBGrid** control is bound to the **RemoteData** control, so it reflects the current set of titles based on the publisher chosen. In the process of populating the result set, the first letter of each title is placed in an array along with the **AbsolutePosition** value for the row. When a letter is chosen and the MoveToRow button is clicked, the **RemoteData** control's **AbsolutePosition** property is set to the value associated with the letter.

```
Dim cn As rdoConnection
Dim en As rdoEnvironment
Dim rs As rdoResultset
Dim LetterIndex() As Long

Private Sub Form_Load()
Dim Li As Integer
'
'   Open the connection. This is a DSN-less Connection
'
Set en = rdoEnvironments(0)
Set cn = en.OpenConnection(dsName:="", _
    Prompt:=rdDriverNoPrompt, _
    Connect:="uid=;pwd=;driver={SQL Server};" _
        & "server=SEQUEL;database=pubs;")

MsRdc1.Connect = cn.Connect

'
'   Fill Publishers list combo box.
'
Set rs = cn.OpenResultset _
    ("Select distinct Pub_Name, Pub_ID" _
   & " from Publishers",  _
   rdOpenStatic, rdConcurReadOnly)
Do Until rs.EOF
    If rs(0) = Null Then
    Else
        PubList.AddItem " " _
         & rs!Pub_ID & ":" & rs!Pub_Name
    End If
    rs.MoveNext
Loop
PubList.ListIndex = 1
rs.Close

PubList_Click ' Make the first query

End Sub

Sub GetLetterIndexes()
'
'  Build an index array for the first
```

```
'  occurance of a letter in the list
'  of titles. Save an AbsolutePosition
'  for each letter.

ReDim LetterIndex(122) As Long
Screen.MousePointer = vbHourglass

Set rs = MsRdc1.Resultset
Do Until rs.EOF
    Li = Convert(Left$(rs!Title, 1))
    If LetterIndex(Li) = 0 Then
        LetterIndex(Li) = rs.AbsolutePosition
    End If
    rs.MoveNext
Loop

Screen.MousePointer = vbDefault
End Sub
'
'  Position the RemoteData control's
'  rdoResultset to the first row of the letter
'  chosen based on the AbsolutePosition
'
Private Sub MoveToRow_Click()
Dim i
i = Convert(LetterWanted)
If LetterIndex(i) > 0 Then
    MsRdc1.Resultset.AbsolutePosition = LetterIndex(i)
Else
    LetterWanted = "(Not Found)"
    Beep
    For i = i + 1 To Asc("z")
        If LetterIndex(i) > 0 Then
            MsRdc1.Resultset.AbsolutePosition = _
      LetterIndex(i)
            LetterWanted = Chr(i + 64)
            Exit For
        End If
    Next i
End If
End Sub

Private Function Convert(Li As String) As Integer
Dim i As Integer
i = Asc(Li) ' Only references first letter
Select Case i
    Case Is < 65: Convert = 0
    Case Is > 122: Convert = 58
    Case Else: Convert = i - 64
End Select
End Function
'
'  Fetch List of Titles for this
'  publisher
'
Private Sub PubList_Click()
Dim PubWanted As String
```

```
'  Pick off the PUB_ID
'
'  Build the SQL Query based on
'  publisher chosen
'
PubWanted = Trim(Left(PubList,
   InStr(PubList, ":") - 1))
Screen.MousePointer = vbHourglass

MsRdc1.SQL = "select * from Titles" _
    & " where Pub_ID = '" _
    & PubWanted & "'" _
    & " order by Title"
MsRdc1.Refresh
Screen.MousePointer = vbDefault
If MsRdc1.Resultset.EOF Then
    MoveToRow.Enabled = False
Else
    MoveToRow.Enabled = True
    GetLetterIndexes
    MsRdc1.Resultset.MoveFirst
End If
End Sub
```

AllowZeroLength Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproAllowZeroLengthC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproAllowZeroLengthA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproAllowZeroLengthS"}

Returns a value that indicates whether a zero-length string ("") is a valid setting for the **Value** property of an **rdoColumn** object with a data type of **rdTypeCHAR**, **rdTypeVARCHAR**, or **rdTypeLONGVARCHAR**.

Syntax

object.**AllowZeroLength**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **AllowZeroLength** property return values are:

Value	Description
True	A zero-length string is a valid value.
False	A zero-length string isn't a valid value.

Remarks

If **AllowZeroLength** is **False** for a column, you must use **Null** to represent "unknown" states — you cannot use empty strings.

AsyncCheckInterval Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproAsyncCheckIntervalC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproAsyncCheckIntervalA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproAsyncCheckIntervalS"}

Returns or sets a value specifying the number of milliseconds that RDO waits between checks to see if an asynchronous query is complete.

Syntax

object.**AsyncCheckInterval** [= *value*]

The **AsyncCheckInterval** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Long expression as described in Remarks.

Remarks

When you use the **rdAsyncEnable** option to execute a query asynchronously, RDO polls the data source periodically to determine if the query has completed. You can change the duration of time between checks by using the **AsyncCheckInterval** property. RDO also checks the status of an asynchronous query when you examine the **StillExecuting** property.

The **AsyncCheckInterval** property defaults to 1000 milliseconds (once a second).

Polling too often can adversely affect both server and workstation performance. Polling less frequently can improve performance, but may affect how quickly data is made available to the user.

As long as the asynchronous query is executing, the **StillExecuting** property returns **True**. Once the query is completed, the **StillExecuting** property is set to false and the QueryComplete event is fired. You can also interrupt and end an asynchronous query by using the **rdoResultset** object's **Cancel** or **Close** method.

Attributes Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproAttributesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproAttributesA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproAttributesS"}

Returns a value that indicates one or more characteristics of an **rdoColumn** object.

Syntax

object.**Attributes**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Attributes** property return value specifies characteristics of the column represented by the **rdoColumn** object and can be a sum of these constants:

Constant	Value	Description
rdFixedColumn	1	The column size is fixed (default for numeric columns) For example, Char, Binary.
rdVariableColumn	2	The column size is variable. For example, VarChar and LongVarChar, VarBinary and LongVarBinary columns.
rdAutoIncrColumn	16	The column value for new rows is automatically incremented to a unique value that can't be changed.
rdUpdatableColumn	32	The column value can be changed.
rdTimeStampColumn	64	The column is a timestamp value. This attribute is set only for **rdClientBatch** cursors.

Remarks

When checking the setting of this property, you should use the **And** operator to test for a specific attribute. Testing for absolute values can jeopardize future compatibility. For example, to determine whether an **rdoColumn** object is fixed-size, you can use code like the following:

```
If MyResultset![ColumnName].Attributes And rdFixedColumn Then...
```

BindThreshold Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBindthresholdC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBindthresholdA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBindThresholdS"}

Returns or sets a value specifying the largest column that will be automatically bound under ODBC.

Syntax

object.**BindThreshold** [= *value*]

The **BindThreshold** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Long expression as described in Remarks.

Remarks

The default value for **BindThreshold** is 1024 bytes.

Several data types support sizes that are far too large to handle using conventional string or byte array techniques. For these columns, you should use the **GetChunk** and **AppendChunk** methods. However, use of these methods is not required — you can simply address the **Value** property assuming the size of the chunk data does not exhaust your resources.

By setting the **BindThreshold** property, you can set the maximum size of chunk that RDO automatically binds to strings. Columns larger than the **BindThreshold** value require use of the **GetChunk** method to retrieve data. The **ChunkRequired** property indicates if the column requires use of **AppendChunk** and **GetChunk** methods by comparing the column's data size against the **BindThreshold** value.

BOF, EOF Properties (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBOFEOFC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBOFEOFA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBOFEOFS"}

- **BOF** — returns a value that indicates whether the current row position is before the first row in an **rdoResultset**.
- **EOF** — returns a value that indicates whether the current row position is after the last row in an **rdoResultset**.

Syntax

object.**BOF**
object.**EOF**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **BOF** property return values are:

Value	Description
True	The current row position is before the first row.
False	The current row position is on or after the first row.

The **EOF** property return values are:

Value	Description
True	The current row position is after the last row.
False	The current row position is on or before the last row.

Remarks

The **BOF** and **EOF** return values are determined by the location of the current row pointer — if this pointer is valid. If either **BOF** or **EOF** is **True**, there is no current row, and any attempt to reference **rdoResultset** data results in a trappable error.

You can use the **BOF** and **EOF** properties to determine whether an **rdoResultset** object contains rows or whether you've gone beyond the limits of an **rdoResultset** as you move from row to row.

If you open an **rdoResultset** containing no rows, **BOF** and **EOF** are set to **True**, and the result set's **RowCount** property setting is 0. When you open an **rdoResultset** that contains at least one row, the first row is the current row and **BOF** and **EOF** are **False**; they remain **False** until you move beyond the beginning or end of the **rdoResultset** using the **MovePrevious** or **MoveNext** method, respectively. When you move beyond the beginning or end of the **rdoResultset**, there is no current row.

If you delete the last remaining row in the **rdoResultset** object, **BOF** and **EOF** might remain **False** until you attempt to reposition the current row.

If you use the **MoveLast** method on an **rdoResultset** containing rows, the last row becomes the current row; if you then use the **MoveNext** method, the current row becomes invalid and **EOF** is set to **True**. Conversely, if you use the **MoveFirst** method on an **rdoResultset** containing rows, the first row becomes the current row; if you then use the **MovePrevious** method, there is no current row and **BOF** is set to **True**.

Typically, when you work with all the rows in an **rdoResultset**, your code will loop through the rows using **MoveNext** until the **EOF** property is set to **True**.

If you use **MoveNext** while **EOF** is set to **True** or **MovePrevious** while **BOF** is set to **True**, a trappable error occurs.

This table shows which *Move* methods are allowed with different combinations of **BOF** and **EOF**.

	MoveFirst, MoveLast	**MovePrevious, Move < 0**	**Move 0**	**MoveNext, Move > 0**
BOF = True, EOF = False	Allowed	Error	Error	Allowed
BOF = False, EOF = True	Allowed	Allowed	Error	Error
Both **True**	Error	Error	Error	Error
Both **False**	Allowed	Allowed	Allowed	Allowed

Allowing a *Move* method doesn't mean that the method will successfully locate a row. It merely indicates that an attempt to perform the specified *Move* method is allowed and won't generate an error. The state of the **BOF** and **EOF** properties may change as a result of the attempted Move.

Effect of specific methods on **BOF** and **EOF** settings:

- An **OpenResultset** method internally invokes a **MoveFirst**. Therefore, an **OpenResultset** on an empty set of rows results in **BOF** and **EOF** being set to **True**.
- All *Move* methods that successfully locate a row set both **BOF** and **EOF** to **False**.
- For dynamic-type **rdoResultset** objects, any **Delete** method, even if it removes the only remaining row from an **rdoResultset**, won't change the setting of **BOF** or **EOF**.
- For other types of **rdoResultset** objects, the **BOF** and **EOF** properties are unchanged as add and delete operations are made because result set membership is fixed.

BOFAction, EOFAction Properties (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBOFActionEOFActionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBOFActionEOFActionA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBOFActionEOFActionS"}

Returns or sets a value indicating what action the **RemoteData** control takes when the **BOF** or **EOF** property is **True**.

Syntax

object.**BOFAction** [= *value*]
object.**EOFAction** [= *value*]

The **BOFAction** and **EOFAction** property syntaxes have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that specifies an action, as described in Settings.

Settings

For the **BOFAction** property, the settings for *value* are:

Constant	Value	Description
rdMoveFirst	0	**MoveFirst** (Default): Keeps the first row as the current row.
rdBOF	1	**BOF**: Moving past the beginning of an **rdoResultset** triggers the **RemoteData** control's Validate event on the first row, followed by a Reposition event on the invalid (**BOF**) row. At this point, the Move Previous button on the **RemoteData** control is disabled.

For the **EOFAction** property, the settings for *value* are:

Constant	Value	Description
rdMoveLast	0	**MoveLast** (Default): Keeps the last row as the current row.
rdEOF	1	**EOF**: Moving past the end of an **rdoResultset** triggers the **RemoteData** control's Validation event on the last row, followed by a Reposition event on the invalid (**EOF**) row. At this point, the Move Next button on the **RemoteData** control is disabled.
rdAddNew	2	**AddNew**: Moving past the last row triggers the **RemoteData** control's Validation event to occur on the current row, followed by an automatic **AddNew**, followed by a Reposition event on the new row.

Remarks

If you set the **EOFAction** property to **rdAddNew**, once the user moves the current row pointer to **EOF** using the **RemoteData** control, the current row is positioned to a new row in the copy buffer. At this point you can edit the newly added row. If you make changes to the new row and the user subsequently moves the current row pointer using the **RemoteData** control, the row is automatically appended to the **rdoResultset**. If you don't make changes to this new row, and reposition the current row to another row, the new row is discarded. If you use the **RemoteData** control to position to another row while it is positioned over this new row, another new row is created.

When you use code to manipulate **rdoResultset** objects created with the **RemoteData** control, the **EOFAction** property has no effect — it only takes effect when manipulating the **RemoteData** control with the mouse.

In situations where the **RemoteData** control **rdoResultset** is returned with no rows, or after the last row has been deleted, using the **rdAddNew** option for the **EOFAction** property greatly simplifies your code because a new row can always be edited as the current row.

Bookmark Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBookmarkC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproLastModifiedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBookmarkA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBookmarkS"}

Returns or sets a bookmark that uniquely identifies the current row in an **rdoResultset** object. If you have a valid bookmark, you can use it to reposition the current row in an **rdoResultset**.

Syntax

object.**Bookmark** [= *value*]

The **Bookmark** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Variant(string) expression that evaluates to a valid bookmark.

Remarks

When a non-forward-only-type **rdoResultset** object is created or opened, each of its rows already has a unique bookmark. You can save the bookmark for the current row by assigning the value of the **Bookmark** property to a variable declared as **Variant**. To quickly return to that row at any time after moving to a different row, set the **rdoResultset** object's **Bookmark** property to the value of that variable.

There is no limit to the number of bookmarks you can establish. To create a bookmark for a row other than the current row, move to the desired row and assign the value of the **Bookmark** property to a **Variant** variable that identifies the row.

To make sure the **rdoResultset** supports bookmarks, inspect the value of its **Bookmarkable** property before you use the **Bookmark** property. If **Bookmarkable** is **False**, the **rdoResultset** doesn't support bookmarks, and using the **Bookmark** property results in a trappable error. While a bookmark value might be returned when using a dynamic cursor, this value cannot always be trusted.

The value of the **Bookmark** property isn't guaranteed to be the same as a row number.

Note The **Bookmark** property doesn't apply to forward-only type **rdoResultset** objects.

Bookmarkable Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBookmarkableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBookmarkableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBookmarkableS"}

Returns a value that indicates whether an **rdoResultset** object supports bookmarks, which you can set using the **Bookmark** property.

Syntax

object.**Bookmarkable**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Bookmarkable** property return values are:

Value	Description
True	The **rdoResultset** supports bookmarks.
False	The **rdoResultset** doesn't support bookmarks.

Remarks

To make sure an **rdoResultset** supports bookmarks, check the **Bookmarkable** property setting before you attempt to set or check the **Bookmark** property.

ChunkRequired Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproChunkRequiredC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproChunkRequiredA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproChunkRequiredS"}

Returns a Boolean value that indicates if data must be accessed using the **GetChunk** method.

Syntax

object.**ChunkRequired**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **ChunkRequired** property return values are:

Value	Description
True	The data should be accessed using the **GetChunk** method.
False	The data need not be accessed using the **GetChunk** method.

Remarks

Use the **ChunkRequired** property to determine if the column in question should be manipulated using the **AppendChunk** and **GetChunk** methods. Accessing the **Value** property of a column whose **ChunkRequired** property is **True**, will *only* result in a trappable error when RDO is unable to fetch the data without use of the **AppendChunk** or **GetChunk** methods. In other words, when the data column does not contain more data than can be handled by conventional string handling, you are *not* required to use the **GetChunk** and **AppendChunk** methods.

By setting the **BindThreshold** property, you can adjust the number of bytes that will force the use of the **AppendChunk** and **GetChunk** methods. You can also determine the length of a *chunk* column by using the **ColumnSize** method.

Connect Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproConnectC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproConnectX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproConnectA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproConnectS"}

Returns or sets a value that provides information about the source of an open **rdoConnection**. The **Connect** property contains the ODBC connect string. This property is always readable, but cannot be changed after the connection is established.

Syntax

object.**Connect** [= *value*]

The **Connect** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression as described in Remarks. (Data type is String)

Settings

The **Connect** property return value is a **String** expression composed of zero or more parameters separated by semicolons, as described in Remarks.

Remarks

When used with the **rdoQuery** or **rdoConnection** objects, this property is read-only unless created as a stand-alone object when it is read-write until the connection is established. The **Connect** property becomes read-write when the **rdoConnection** object is closed. When used with the **RemoteData** control, this property is read-write.

The **Connect** property is used to pass additional information to and from the ODBC driver manager to establish a connection with a data source. The **Connect** property holds the ODBC connect string which is also used as an argument to the **OpenConnection** method. When used with a stand-alone **rdoConnection** or **rdoQuery** objects, the **Connect** property is used by the **EstablishConnection** method.

Except when associated with the **RemoteData** control, once a connection is made, the **Connect** property is completed with the values optionally supplied by the user and the ODBC driver manager. The **Connect** property of the **rdoQuery** contains this amended connect string.

The **RemoteData** control's **Connect** property is not changed after the connection is established. However, the completed connect string can be extracted from the **RemoteData** control's **Connection** property. For example:

```
FullConnect = MSRDC1.Connection.Connect
```

The following table details valid ODBC connect string arguments and typical usage. Note that each parameter is delineated with a semi-colon (;).

ODBC Connect String Arguments

Parameter	Specifies	Example
DSN	Registered ODBC data source by name.	`DSN=MyDataSource;` (If specified when establishing a DSN-less connection, DSN must be the *last* argument)

UID	User name of a recognized user of the database	`UID=Victoria;`
PWD	Password associated with user name	`PWD=ChemMajor;`
DRIVER	Description of driver. (Note brackets for driver names that include spaces.)	`DRIVER={SQL Server};`
DATABASE	Default database to use once connected	`DATABASE=Pubs;`
SERVER	Name of remote server	`SERVER=SEQUEL;`
WSID	Workstation ID (your system's Net name)	`WSID=MYP5`
APP	Application name. At design time this is set to your project name. At runtime this is your .exe name.	`APP=Accounting`

Note Some ODBC drivers require different parameters not shown in this list.

For example, to set the **Connect** property of a **RemoteData** control you could use code like the following:

```
Dim Cnct As String
Cnct = "DSN=WorkData;UID=Chrissy;" _
      & "PWD=MIDFLD;DATABASE=WorkDB;"
RemoteData1.Connect = Cnct
RemoteData1.SQL = "Select Name, City " _
   & " From Teams Where Type = 12"
RemoteData1.Refresh
```

You can use this same connect string to establish a new connection:

```
Dim Cn As rdoConnection
Set Cn = rdoEnvironments(0).OpenConnection("", _
rdDriverNoPrompt,True,Cnct$)
```

Note Valid parameters are determined by the ODBC driver. The parameters shown in the preceding example are supported by the Microsoft SQL Server ODBC driver. ODBC, LOGINTIMEOUT and DBQ are not valid parameters of the **RemoteData** control or the **rdoConnection** object's **Connect** property. These parameters are supported by the Microsoft Jet database engine, and not by the ODBC driver. To set login timeout delay, you must use the **LoginTimeout** property of the **rdoEnvironment** object.

Capturing Missing Arguments

If the connect string is null, the information provided by the DSN is incomplete, or invalid arguments are provided, the connection cannot be established. If your code sets the *prompt* argument of the **OpenConnection** method or the **RemoteData** control's **Prompt** property to prohibit user completion of missing ODBC connect arguments, a trappable error is triggered. Otherwise the ODBC driver manager displays a dialog box to gather missing information from the user. Depending on the setting of the **Prompt** argument of the **OpenConnection** or **EstablishConnection** methods, these dialogs capture the DSN from a list of registered ODBC data sources. Names presented to the user, and optionally, the user ID and password. If the connection fails to succeed using these user-provided values, the dialogs are presented again until the connection succeeds or the user cancels the operation. In some cases, the user can create their own DSN using these dialogs.

If a password is required, but not provided in the **Connect** property setting, a login dialog box is displayed the first time a table is accessed by the ODBC driver and each time the connection is

closed and reopened.

Connecting with Domain-Managed Security

When connecting to ODBC data sources that support domain-managed security, set the UID and PWD parameters to "". In this case, the Windows NT user name and password are passed to the data source for validation. This strategy permits access to the data source by users with access to the NT domain through authenticated workstation logons.

You can set the **Connect** property for an **rdoConnection** object by providing a *connect* argument to the **OpenConnection** method. Once the connection is established, you can check the **Connect** property setting to determine the DSN, database, user name, password, or ODBC data source of the database.

Registering Data Source Names

Before you can establish a connection using a Data Source Name (DSN), you must either manually register the DSN using the Windows control panel application or use the **rdoRegisterDataSource** method. This process establishes the server name, driver name and other options used when referencing this data source.

Establishing DSN-Less Connections

Under the right circumstances you might not need to pre-register a DSN before connecting. If the following conditions are met, RDO can establish a DSN-less connection using the **RemoteData** control, or the **OpenConnection** or **EstablishConnection** methods with a fully-populated **Connect** property or connect string:

- The connection uses the default named-pipes networking protocol.
- The connection does not set the OEMTOANSI option.
- You specify the name of the server using the SERVER argument in the connect string.
- You specify the name of the ODBC driver using the DRIVER argument in the connect string.
- You set the DSN argument in the connect string (or wherever it appears — as in the **DataSourceName** property of the **RemoteData** control) to an empty string. The empty DSN argument must be specified as the *last* parameter of the connect string.

Connect Property and OpenConnection Example: DSN-Less Connection Using OpenConnection

The following example establishes a DSN-less ODBC connection using the **OpenConnection** method against the default **rdoEnvironment**. In this case the example prints the resulting **Connect** property to the Immediate window.

```
Dim en as rdoEnvironment
Dim cn as rdoConnection

Set en = rdoEnvironments(0)
Set cn = en.OpenConnection(dsName:="", _
    Prompt:=rdDriverNoPrompt, _
    Connect:="uid=;pwd=;driver={SQL Server};" _
        & "server=SEQUEL;database=pubs;")
debug.print cn.Connect
```

Connect Property and OpenConnection Example: DSN Connection Using OpenConnection

The following example establishes an ODBC connection using the **OpenConnection** method but requires the user to provide all connection information. In this case the example prints the resulting **Connect** property to the Immediate window.

```
Dim cn As rdoConnection
Dim en As rdoEnvironment

Set en = rdoEnvironments(0)
Set cn = en.OpenConnection(dsName:="WorkDB", _
    Prompt:=rdDriverCompleteRequired)
debug.print cn.Connect
```

Connect Property Example: DSN-Less Connection Using Stand-Alone Connection

The following example establishes a DSN-less ODBC connection by creating a stand-alone **rdoConnection** object and uses the **EstablishConnection** method to open the connection. Note that the `DSN='';` argument is positioned at the end of the connect string. The example prints the resulting **Connect** property to the Immediate window.

```
' Create a DSN-less connection
'  using a stand-alone rdoConnection object and
'  the EstablishConnection method
'
Dim cn As New rdoConnection
Dim qd As New rdoQuery

cn.Connect = "uid=;pwd=;server=SEQUEL;" _
    & "driver={SQL Server};database=pubs;" _
    & "DSN='';"
cn.cursordriver = rdUseOdbc
cn.EstablishConnection rdDriverNoprompt
debug.print cn.Connect

Set qd.ActiveConnection = cn
```

Connect Property Example: DSN Connection Using Establish Connection

The following example establishes an ODBC connection using a registered DSN to provide most of the required arguments. The User ID and Password are to be provided by domain-managed security. In this case the example prints the resulting **Connect** property to the Immediate window.

```
Dim cn As New rdoConnection
Dim qd As New rdoQuery

cn.Connect = "uid=;pwd=;"DSN=WorkDB;"
cn.cursordriver = rdUseOdbc
cn.EstablishConnection rdDriverNoprompt
debug.print cn.Connect
```

Connection Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproConnectionA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproConnectionS"}

Returns a reference to a **RemoteData** control's underlying **rdoConnection** object.

Syntax

object.**Connection**
Set *connection* **=** *object*.**Connection**

The **Connection** property syntax has these parts:

Part	Description
connection	An object expression that evaluates to a valid **rdoConnection** object.
object	An object expression that evaluates to an object in the Applies To list.

Remarks

When a **RemoteData** control is initialized, **RemoteData** opens a connection to the data source specified in the control's **Connect** property. The **rdoConnection** object created by RDO is exposed by the **Connection** property.

rdoConnection objects have properties and methods you can use to manage data. You can use any method of an **rdoConnection** object, such as **Close** and **Execute**, with the **Connection** property of a **RemoteData** control.

Except when associated with the **RemoteData** control, once a connection is made, the **Connect** property is completed with the values optionally supplied by the user and the ODBC driver manager. The **Connect** property of the **rdoQuery** contains this amended connect string.

The **RemoteData** control's **Connect** property is not changed after the connection is established. However, the completed connect string can be extracted from the **RemoteData** control's **Connection** property. For example:

```
FullConnect = MSRDC1.Connection.Connect
```

CursorDriver Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproCursorDriverC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproCursorDriverA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproCursorDriverS"}

Returns or sets a value that specifies the type of cursor to be created.

Syntax

object.**CursorDriver** [= *value*]

The **CursorDriver** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An **Integer** or constant as described in Settings.

Settings

Constant	Value	Description
rdUseIfNeeded	0	The ODBC driver will choose the appropriate style of cursors. Server-side cursors are used if they are available.
rdUseOdbc	1	**RemoteData** will use the ODBC cursor library.
rdUseServer	2	Use server-side cursors.
rdUseClientBatch	3	RDO will use the optimistic batch cursor library.
rdUseNone	4	Result set is not returned as a cursor.

Remarks

The **CursorDriver** property only affects connections established *after* the **CursorDriver** property has been set — the property is read-only on existing connections.

When the initial (default), and each subsequent **rdoEnvironment** object is created, the **CursorDriver** property is set from the **rdoEngine** object's **rdoDefaultCursorDriver** property which is set using the same constants.

Choosing a Cursor Driver

Choosing the right cursor driver can have a significant impact on the overall performance of your application, what resources are consumed by the cursor, and limit the type or complexity of the cursors you create. Each type of cursor has its own benefits and limitations. In many cases, the best choice is no cursor at all because your application often does not need to scroll through the data or perform update operations against a keyset.

The following paragraphs outline the functionality and suggested purposes for each of the cursor types.

- Server-Side Cursors

  This cursor library maintains the cursor keyset on the server (in *TempDB*) which eliminates the need to transmit the keyset to the workstation where it consumes needed resources. However, this cursor driver consumes *TempDB* space on the remote server so this database must be expanded to meet this requirement. Cursors created with the server-side driver cannot contain

more than one SELECT statement – if they do, a trappable error is fired. You can still use the server-side cursor driver with multiple result set queries if you disable the cursor by creating a forward-only, read-only cursor with a rowset size of one. Not all remote servers support server-side cursors. Note that server-side cursors are enabled when using either **rdUseIfNeeded** or **rdUseServer** against Microsoft SQL Server databases.

- ODBC Client-Side Cursors

  This cursor library builds keysets on the workstation in local RAM overflowing to disk if necessary. Because of this design considerably more network operations must be performed to initially create the keyset, but with small cursors this should not impose a significant load on the workstation or network. ODBC client-side cursors do not impose any type of restriction on the type of query executed. This option gives better performance for small result sets, but degrades quickly for larger result sets.

- Client-Batch Cursors

  This cursor library is designed to deal with the special requirements of optimistic batch updates and several other more complex cursor features. Client-batch cursors are required for dissociate connections, batch mode, and multi-table updates. This cursor also supports delayed BLOB column fetch, buffered cursors, and additional control over updates. This library is somewhat larger than the others, but also performs better in many situations.

- The No-Cursor Option

  In cases where you need to fetch rows quickly, or perform action queries against the database without the overhead of a cursor, you can choose to instruct RDO to bypass creation of a cursor. Basically, this option creates a forward-only, read-only result set with a **RowsetSize** set to 1. This option can improve performance in many operations. While you cannot update rows or scroll between rows with this cursor, you can submit independent action queries to manipulate data. This option is especially useful when accessing data through stored procedures.

DataSourceName Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproDataSourceNameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproDataSourceNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproDataSourceNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproDataSourceNameS"}

Returns or sets the data source name for a **RemoteData** control.

Syntax

object.**DataSourceName** [= *datasourcename*]

The **DataSourceName** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
datasourcename	A string expression that indicates a registered data source name.

Remarks

This property can be left blank if the **RemoteData** control's **Connect** property identifies a data source name (DSN) registered in the Windows Registry (32-bit) or if you create a DSN-less connection that provides all required information in the **Connect** property.

Once the **rdoConnection** is opened by the **RemoteData** control, the **DataSourceName** property contains the DSN used to establish the connection — it may be different from the value set before the connection is opened, because a user might select a data source from a list of valid DSN entries during the connection process.

If you change this property after the control's **rdoConnection** object is open, you must use the **RemoteData** control's **Refresh** method to open a new connection to the data source.

Description Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproDescriptionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproNumberX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproDescriptionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproDescriptionS"}

Returns a descriptive string associated with an error.

Syntax

object.**Description**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Description** property return value is a string expression containing a description of the error.

Remarks

When an error occurs either on the remote server, or in the ODBC interface while processing your query, an **rdoError** object is created and appended to the **rdoErrors** collection. The **rdoError** object's **Description** property returns a short description and context information about where the error occurred. This can be used to alert the user to an error that you cannot, or do not want to handle. The SQLState code is appended to the front of message, followed by a colon and a space. For example "S0021: Cannot find XXX".

Direction Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproDirectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproDirectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproDirectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproDirectionS"}

Returns or sets a value indicating how a parameter is passed to or from a procedure.

Syntax

object.**Direction** [= *value*]

The **Direction** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or **Integer** as described in Settings.

Settings

The settings for *value* are one of the following values:

Constant	Value	Description
rdParamInput	0	(Default) The parameter is used to pass information to the procedure.
rdParamInputOutput	1	The parameter is used to pass information both to and from the procedure.
rdParamOutput	2	The parameter is used to return information from the procedure as in an output parameter in SQL.
rdParamReturnValue	3	The parameter is used to return the return status value from a procedure.

Remarks

When working with stored procedures, and parameter queries you should identify those parameters that are to be managed by RDO on your behalf — but only when using drivers that do not automatically detect parameter direction. A parameterized query can take virtually any number of input arguments — each of these need to be marked when you create your query.

Generally, your query returns a set of rows that meet the requirements established in the query based on the parameters you provide at runtime. However, when working with stored procedures, another aspect is exposed. Stored procedures return information using row sets, return status, and output parameters. Because of this, each parameter returned by your stored procedure must be marked when creating your query.

The **Direction** property determines whether the parameter is an input parameter, output parameter, or both — or if the parameter is the return value from the procedure.

Note When first addressing the rdoParameter object to set the Direction property you might trip a trappable error if the rdoParameters collection could not be created. Generally this is due to syntax errors in the query or other problems that prevented RDO from creating the collection.

Some ODBC drivers do not provide information on the direction of parameters to a SELECT statement or procedure call so all parameter directions default to **rdParamInput.** In these cases, it is necessary to set the direction in code prior to executing the query.

Note The Microsoft SQL Server 6.x driver automatically sets the **Direction** property for all procedure parameters so you should not have to set the **Direction** property for any of your queries' parameters.

The **Direction** property is associated with the **rdoParameter** object but it is generally unnecessary to address the **rdoParameter** object itself as it is the default collection of the **rdoQuery** object as shown in the examples below.

For example, the following procedure returns a value from a stored procedure:

```
{? = call sp_test}
```

This call produces one parameter — the return value. It is necessary to set the direction of this parameter to **rdParamOutput** or **rdParamReturnValue** before executing the prepared statement. For example:

```
Dim my_statement As rdoQuery
Set my_statement = someRdoConnection.CreateQuery _
   ("MyPs", "{? = call sp_testprocedure }", ...)
my_statement.rdoParameters(0).Direction =  _
   rdParamReturnValue
my_statement.Execute
Print my_statement.rdoParameters(0)
```

You need to set all parameter directions except **rdParamInput** before accessing or setting the values of the parameters and before executing the **rdoQuery**.

You should use **rdParamReturnValue** for return values, but you can use **rdParamOutput** where **rdParamReturnValue** is not supported.

EditMode Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproEditModeA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproEditModeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproEditModeS"}

Returns a value that indicates the state of editing for the current row.

Syntax

object.**EditMode**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **EditMode** property returns an Integer or constant as described in the following table:

Constant	Value	Description
rdEditNone	0	No editing operation is in progress.
rdEditInProgress	1	The **Edit** method has been invoked, and the current row is in the copy buffer.
rdEditAdd	2	The **AddNew** method has been invoked, and the current row in the copy buffer is a new row that hasn't been saved in the database.

Remarks

The **EditMode** property is most useful when you want to depart from the default functionality of a **RemoteData** control. You can check the value of the **EditMode** property and the value of the *action* parameter in the Validate event procedure to determine whether to invoke the **UpdateRow** method.

You can also check to see if the **LockEdits** property of the **rdoResultset** is **True** and the **EditMode** property setting is **rdEditInProgress** to determine whether the current data page is locked.

Environment Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproEnvironmentC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproEnvironmentA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproEnvironmentS"}

Returns a reference to a **RemoteData** control's underlying **rdoEnvironment** object.

Syntax

object.**Environment**
Set *environment* **=** *object*.**Environment**

The **Environment** property syntax has these parts:

Part	Description
environment	An object expression that evaluates to a valid **rdoEnvironment** object.
object	An object expression that evaluates to an object in the Applies To list.

Remarks

When a **RemoteData** control is initialized, **RemoteData** uses the default **rdoEnvironments**(0) — the **Environment** property is initially set to this object.

If you assign another **rdoResultset** to the **RemoteData** control's **Resultset** property, the **Environment** property is set to the **rdoEnvironment** object used to create the result set.

rdoEnvironment objects have properties and methods you can use to manage data. For example, you can use any method of an **rdoEnvironment** object, such as **OpenConnection**, **BeginTrans**, **CommitTrans**, or **RollbackTrans**, with the **Environment** property.

ErrorThreshold Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproErrorthresholdC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproErrorthresholdA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproErrorthresholdS"}

Returns or sets a value that determines the severity level that constitutes a fatal error.

Note This property is provided for backward compatibility with RDO version 1.0 code. It should be replaced with code that implements the **rdoEngine** object's InfoMessage event which provides equivalent functionality.

Remarks

In version 4.x of Microsoft SQL Server, it is not possible to set the severity of errors using the RAISERROR statement. As a result, the ErrorThreshold property was needed to permit your code to filter those messages beyond a threshold of severity.

Version 6.x of Microsoft SQL Server now supports the inclusion of a severity level in the RAISERROR statement so it is no longer necessary to use the ErrorThreshold property.

All errors that are returned with a severity of less than 10 are trapped by the ODBC layers and set the SQL_SUCCESS_WITH_INFO result code. This causes RDO to raise the InfoMessage event but not stop query processing.

HelpContext, HelpFile Properties (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproHelpContextHelpFileC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproHelpContextHelpFileA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproHelpContextHelpFileS"}

- **HelpContext** — returns a context ID for a topic in a Microsoft Windows Help file.
- **HelpFile** — returns a fully qualified path to the Help file as a variable.

Syntax

object.**HelpContext**
object.**HelpFile**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **HelpContext** property returns a **Long** value.

The **HelpFile** property returns a **String** value.

Remarks

If a Microsoft Windows Help file is specified in **HelpFile**, the **HelpContext** property is used to automatically display the Help topic it identifies.

Note You should write routines in your application to handle typical errors. When programming with an object, you can use the Help supplied by the object's Help file to improve the quality of your error handling, or to display a meaningful message to your user if the error is not recoverable.

hDbc Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprohDbcC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdprohDbcX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprohDbcA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprohDbcS"}

Returns a value corresponding to the ODBC connection handle.

Syntax

object.**hDbc**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **hDbc** property returns a Long value containing the ODBC connection handle created by the ODBC driver manager corresponding to the specified **rdoConnection** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hDbc** connection handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

If your application requires access to special ODBC connection option settings, these should be set or retrieved using the **hDbc** property *before* the connection is established. Resetting ODBC settings of any kind after the connection is established can result in unpredictable behavior.

hDbc Property Example

The following example illustrates use of the hDbc property when executing an ODBC API function. In this case, the application sets a connection option that changes how transactions are isolated.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As rdoConnection
Dim rc As Integer

'Declare Function SQLSetConnectOption Lib "odbc32.dll" (ByVal hdbc&, ByVal
fOption%, ByVal vParam As Any) As Integer
'
'Transaction isolation option masks
'
 Const SQL_TXN_ISOLATION As Long = 108
 Const SQL_TXN_READ_UNCOMMITTED As Long = &H1&
 Const SQL_TXN_READ_COMMITTED As Long = &H2&
 Const SQL_TXN_REPEATABLE_READ As Long = &H4&
 Const SQL_TXN_SERIALIZABLE As Long = &H8&
 Const SQL_TXN_VERSIONING As Long = &H10&

Private Sub Form_Load()
Set en = rdoEngine.rdoEnvironments(0)

Set cn = en.OpenConnection(dsName:="WorkDB", _
    Prompt:=rdDriverNoPrompt, _
    Connect:="Uid=;pwd=;database=workdb")

rc = SQLSetConnectOption(cn.hDbc, SQL_TXN_ISOLATION,
SQL_TXN_READ_UNCOMMITTED)

Debug.Print rc

End Sub
```

hEnv Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprohEnvC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdprohEnvX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprohEnvA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprohEnvS"}

Returns a value corresponding to the ODBC environment handle.

Syntax

object.**hEnv**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **hEnv** property returns a Long value containing the ODBC environment handle created by the ODBC driver manager corresponding to the specified **rdoEnvironment** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hEnv** environment handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

hEnv Property Example

The following example illustrates use of the **hEnv** property when accessing an ODBC API function. This code displays all registered data source names (DSNs) in a **ListBox** control.

```
Private Sub ShowDSNs_Click()
Dim fDirection As Integer
Dim szDSN As String * 1024
Dim cbDSNMax As Integer
Dim pcbDSN As Integer
Dim szDescription As String * 1024
Dim cbDescriptionMax As Integer
Dim pcbDescription As Integer
Dim Item As String
Set En = rdoEnvironments(0)
fDirection = SQL_FETCH_NEXT
cbDSNMax = 1023
cbDescriptionMax = 1023
List1.Clear
I = SQL_SUCCESS
 While I = SQL_SUCCESS
     szDSN = String(1024, " ")
     szDescription = String(1024, " ")
     I = SQLDataSources(En.hEnv, fDirection, szDSN, _
     cbDSNMax, pcbDSN, szDescription, _
     cbDescriptionMax, pcbDescription)
     Item = Left(szDSN, pcbDSN) & " - "  _
      & Left(szDescription, pcbDescription)
     Debug.Print Item
     List1.AddItem Item
 Wend

End Sub
```

hStmt Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprohStmtC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdprohStmtX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprohStmtA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprohStmtS"}

Returns a value corresponding to the ODBC statement handle.

Syntax

object.**hStmt**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **hStmt** property returns a Long value containing the ODBC statement handle created by the ODBC driver manager corresponding to the specified **rdoResultset** object.

Remarks

This handle can be used to execute ODBC functions that require an ODBC **hStmt** statement handle.

Note While it is possible to execute ODBC API functions using the ODBC **hEnv, hDbc**, and **hStmt** handles, it is recommended that you do so with caution. Improper use of arbitrary ODBC API functions using these handles can result in unpredictable behavior. You should not attempt to save this handle in a variable for use at a later time as the value is subject to change.

hStmt Property Example

This example illustrates use of the **hStmt** property to return a configuration option for a specific statement handle. The example uses the **SQLGetStmtOption** function to determine the type of cursor created by the **OpenResultset** method. Note that this value is also supplied by the **rdoResultset Type** property.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As rdoConnection
Dim rs As rdoResultset
Dim rc As Integer
Dim CursorType As Long
Dim T As String

'Declare Function SQLGetStmtOption Lib "odbc32.dll" (ByVal hstmt&, ByVal
fOption%, ByRef pvParam As Any) As Integer

Private Sub Form_Load()
Set en = rdoEngine.rdoEnvironments(0)

en.CursorDriver = rdUseOdbc

Set cn = en.OpenConnection(dsName:="WorkDB", _
    prompt:=rdDriverNoPrompt, _
    Connect:="Uid=;pwd=;database=Pubs")

Set rs = cn.OpenResultset("Select * from Publishers", _
 rdOpenKeyset, rdConcurRowVer)

Select Case rs.Type
    Case rdOpenForwardOnly: T = "Forward-only"
    Case rdOpenStatic: T = "Static"
    Case rdOpenKeyset: T = "Keyset"
    Case rdOpenDynamic: T = "Dynamic"
End Select
MsgBox "RDO indicates that a " & T  _
   & " Cursor was created"
CursorType = 0
rc = SQLGetStmtOption(rs.hStmt,  _
   SQL_CURSOR_TYPE, CursorType)

Select Case CursorType
    Case SQL_CURSOR_FORWARD_ONLY: T = "Forward-only"
    Case SQL_CURSOR_STATIC: T = "Static"
    Case SQL_CURSOR_KEYSET_DRIVEN: T = "Keyset"
    Case SQL_CURSOR_DYNAMIC: T = "Dynamic"
End Select
MsgBox "ODBC indicates that a " & T  _
   & " Cursor was created"
End Sub
```

KeysetSize Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproKeysetsizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproKeysetsizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproKeysetsizeS"}

Returns or sets a value indicating the number of rows in the keyset buffer.

Syntax

object.**KeysetSize** [= *value*]

The **KeysetSize** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** expression as described in Settings.

Settings

The settings for *value* must be greater than or equal to the **RowsetSize** property.

Remarks

The **KeysetSize** property is a value that specifies the number of rows in the keyset for a keyset- or dynamic-type **rdoResultset** cursor. If the keyset size is 0 (the default), the cursor is fully keyset-driven. If the keyset size is greater than 0, the cursor is mixed (keyset-driven within the keyset and dynamic outside the keyset).

If **KeysetSize** is a value greater than **RowsetSize**, the value defines the number of rows in the keyset that are to be buffered by the driver.

Not all ODBC data sources support keyset cursors.

Note Because version 2.5 of the Microsoft SQL Server ODBC driver does not support mixed-style cursors, if you set a *value*, **KeysetSize** is reset to 0 and the driver returns error 01S02: "`Option value changed.`"

Warning When using **rdConcurLock** concurrency (pessimistic), the **KeysetSize** determines the number of rows locked when the cursor is first opened. The entire keyset remains locked as long as the cursor remains open.

LastModified Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLastModifiedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproLastModifiedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLastModifiedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLastModifiedS"}

Returns a bookmark indicating the most recently added or changed row.

Syntax

object.**LastModified**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The return value for this property is a **Variant**(string) data type, as described in Remarks.

Remarks

After you use the **AddNew** method to add a new row, or edit an existing row using the **Update** method, the **LastModified** property returns a bookmark as a pointer to the row most recently modified — providing the keyset supports additions. That is, if new rows are added to the keyset as well as the underlying database table(s), the **LastModified** property will point to this new row in the keyset.

To position the current row pointer to this row, set the **Bookmark** property of the (same) **rdoResultset** object to the **LastModified** property.

If there have been no modifications against this **rdoResultset**, then the **LastModified** property returns 0.

Not all types of **rdoResultsets** support additions to their keysets so the **LastModified** property might return 0 after a row has just been added. For example, while a ODBC cursor static keyset can be updated, its rowset cannot be added to. Inserts are performed on the database, but not to the keyset, so the **LastModified** property always returns 0 in this case.

Server-side keyset cursors support additions to the keyset so the **AddNew** method sets the **LastModified** property — as does the **Update** method. .

Client-side static cursors do not add new rows to the cursor's membership, thus the **LastModified** property value is undefined when using the **AddNew** method. However, it is defined after the **Update** method is used. Server-side static cursors are read-only, so the **LastModified** property is not relevant in this case.

Dynamic and forward-only cursors do not support bookmarks (as indicated by the **Bookmarkable** property returning False), so the **LastModified** property is not relevant in these cursors.

The client batch cursor library also supports the **LastModified** property. For static cursors, new rows are added to the cursor membership so the **AddNew** method sets the **LastModified** property — as does the **Update** method.

LastModified, Bookmark Properties Example

This example illustrates use of the **LastModified** property to reposition the current row pointer to the row most recently modified by RDO. The code opens a connection against SQL Server and creates a keyset cursor-based query on the Authors table. The query expects a single parameter to pass in the name of the Author to edit. Once selected, edited and updated, the row pointer is repositioned to the last row modified by setting the bookmark property of the **rdoResultset** to the **LastModified** property.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim col As rdoColumn

Private Sub TestLM_Click()
qy(0) = LookFor.Text

rs.Edit
rs!City = NewCity.Text  ' a TextBox control
rs.Update

rs.Bookmark = rs.LastModified

'Simply show data in picture control
Pic.Cls  'Clear the picture control.

For Each col In rs.rdoColumns
    Pic.Print col.Name,
Next
Pic.Print String(80, "-")
For Each col In rs.rdoColumns
    Pic.Print col,
Next

End Sub

Private Sub Form_Load()
cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=sequel;" _
   & "driver={SQL Server};database=pubs;dsn='';"
cn.EstablishConnection

With qy
    .Name = "ShowWhite"
    .SQL = "Select * from Authors " _
      & " where Au_LName like ?"
    .LockType = rdConcurReadOnly
    .CursorType = rdOpenForwardOnly
    .RowsetSize = 1
    Set .ActiveConnection = cn
End With

qy(0) = LookFor.Text    ' a textbox control
Set rs = qy.OpenResultset(rdOpenKeyset, rdConcurRowver)
```

```
Exit Sub
End Sub
```

LockType Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLockTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLockTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLockTypeS"}

Returns or sets a Long integer value indicating the type of concurrency handling.

Syntax

object.**LockType** [= *value*]

The **LockType** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or **Long** value as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
rdConcurReadOnly	1	(Default) Cursor is read-only. No updates are allowed.
rdConcurLock	2	Pessimistic concurrency.
rdConcurRowVer	3	Optimistic concurrency based on row ID.
rdConcurValues	4	Optimistic concurrency based on row values.
rdConcurBatch	5	Optimistic concurrency using batch mode updates. **Status** values returned for each row successfully updated.

Remarks

In order to maintain adequate control over the data being updated, RDO provides a number of concurrency options that control how other users are granted, or refused access to the data being updated. In many cases, when you lock a particular row using one of the **LockType** settings, the remote engine might also lock the entire page containing the row. If too many pages are locked, the remote engine might also escalate the page lock to a table lock to improve overall system performance.

Not all lock types are supported on all data sources. For example, for SQL Server and Oracle servers using the **rdUseODBC** cursor library, static-type **rdoResultset** objects can only support **rdConcurValues** or **rdConcurReadOnly**.

If the concurrency option is not supported by the data source, the driver substitutes a different concurrency option at execution time if one is available. If the driver cannot substitute a suitable alternative concurrency option, a trappable error is fired (SQLState Code 01S02 `"Option Value Changed"`). For **rdConcurValues**, the driver substitutes **rdConcurRowVer** and vice versa. For **rdConcurLock**, the driver substitutes, in order: **rdConcurRowVer** or **rdConcurValues**.

Choosing a Concurrency Option

Note RDO concurrency does not function as it does with Data Access Objects (DAO). Be sure to

review the following sections to determine the best type of concurrency control for your application.

- **Read-Only Concurrency:** This option does not impose any exclusive locks on the rows fetched. In most cases, however, you must be granted a share lock to gain access to the rows. In other words, other users cannot have exclusive locks (read-write or intend to write locks) on the pages being accessed. Choosing this option makes the cursor read-only. This does not preclude use of action queries to update the data independent of the cursor. This is the default **LockType**.
- **Pessimistic Concurrency:** This option requests an *immediate* exclusive lock on the cursor rows which implements the lowest level of locking sufficient to ensure the row can be updated. Unlike DAO, which defers locking until the **Edit** method is used, RDO locks the first **RowsetSize** rows of the result set when the cursor is first opened with the **OpenResultset** method. That is, if your **RowsetSize** is 100 rows, the remote engine is instructed to lock each page that contains one of these selected rows. This means up to 100 pages can be locked – which can lock hundreds of rows. As the current row pointer is moved through the result set, additional pages are locked, and those no longer referenced are released. This technique assures your application that no other application is granted exclusive (read-write) access to any rows being processed by the cursor.
- **Optimistic Concurrency:** This type of concurrency management does not lock any rows or pages – it simply compares the row being posted to the database with the row as it currently exists on the server. Depending on the type of optimistic concurrency chosen, RDO and the ODBC layers compare either the row ID, the row data values, TimeStamp columns or combinations of these options with existing data to determine if a row has changed since last fetched. If no changes have taken place since the last fetch, the update is made. Otherwise, your application triggers a trappable error.

The **LockType** property supports three types of optimistic concurrency as described below. When using the Optimistic Batch Concurrency option (**rdConcurBatch**), you should also set the **UpdateCriteria** property to choose an appropriate update concurrency option.

- **Optimistic Concurrency – Row Version:** By comparing the row identifier (usually a TimeStamp column) , RDO can determine if the row has changed since last fetched. If it has, a trappable error results.
- **Optimistic Concurrency – Row Values:** By comparing row values on a column-by-column basis, RDO can determine if the row has changed since last fetched. If it has, a trappable error results.
- **Optimistic Batch:** This type of concurrency uses the **UpdateCriteria** property to determine how to test if rows have changed when using the **UpdateBatch** method.

LockEdits Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLockEditsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLockEditsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLockEditsS"}

Returns a Boolean value indicating the type of locking that is in effect.

Syntax

object.**LockEdits**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The return values for **LockEdits** are:

Setting	Description
True	Pessimistic locking is in effect.
False	(Default) Optimistic locking is in effect.

Remarks

If a page is locked and the data source uses page locking, no other user can edit rows on the same page. If row-level locking is used, the row being edited and all other rows in the rowset are locked. The rowset is defined as the number of rows specified by the **RowsetSize** property. If **LockEdits** is **True** and another user already has the page locked, an error occurs when you use the **OpenResultset** method. Generally, other users can read data from locked pages.

If **LockEdits** is **False** (the default) and you later use **Update** while the page is locked by another user, an error occurs. To see the changes made to your row by another user (and lose your changes), set the **Bookmark** property of your **rdoResultset** object to itself.

Note Data page size is determined by the data source. Microsoft SQL Server uses 2K data pages.

LoginTimeout Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLoginTimeoutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLoginTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLoginTimeoutS"}

Returns or sets a value that specifies the number of seconds the ODBC driver manager waits before a timeout error occurs when a connection is opened.

Syntax

object.**LoginTimeout** [= *value*]

The **LoginTimeout** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the number of seconds the driver manager waits before timing out and returning an error.

Remarks

If *value* is 0, no timeout occurs and an error does not occur if a connection cannot be established. If you are not using asynchronous connections, this might cause your application to block indefinitely.

When you're attempting to connect to an ODBC database, such as SQL Server, there may be delays due to network traffic or heavy use of the ODBC data source. Rather than waiting indefinitely, you can specify how long to wait before the ODBC driver manager produces an error.

The default timeout value is either 15 seconds or a value set by the **rdoDefaultLoginTimeout** property. When used with an **rdoEnvironment** object, the **LoginTimeout** property specifies a global value for all login operations associated with the **rdoEnvironment**. The **LoginTimeout** setting of on an **rdoConnection** object overrides the default value.

If the specified timeout exceeds the maximum timeout in the data source, or is smaller than the minimum timeout, the driver substitutes that value and the following error is logged in the **rdoErrors** collection: SQLState 01S02 `"Option value changed."`

Typically, a connection to a remote server on a Local Area Network (LAN) takes under eight seconds to complete. Remote Access Service (RAS) or Internet connections can take far longer depending on Wide Area Network bandwidth, load and other factors.

LogMessages Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLogmessagesC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLogmessagesA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLogmessagesS"}

Enables ODBC trace logging and returns or sets a value indicating the path of the ODBC trace file created by the ODBC driver manager to record all ODBC operations.

Syntax

object.**LogMessages** [= *value*]

The **LogMessages** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **String** expression as described in Settings.

Settings

V*alue* contains the path of an ASCII file used to log ODBC operations. If the **LogMessages** property is an empty string, no logging takes place.

Remarks

When the **LogMessages** property is **True**, all ODBC commands are sent to an ASCII log file that can be used to debug or tune queries or other operations.

On Windows NT or Windows 95, tracing should *only* be used for a single application or each application should specify a different trace file. Otherwise, two or more applications might attempt to open the same trace file at the same time, causing an error.

Note ODBC performance is adversely affected when the log is enabled.

MaxRows Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproMaxrowsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproMaxrowsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproMaxrowsS"}

Returns or sets a value indicating the maximum number of rows to be returned from a query or processed in an action query.

Syntax

object.**MaxRows** [= *value*]

The **MaxRows** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** expression as described in Settings.

Settings

The setting for *value* ranges from 0 to any number. If *value* is set to 0, no limit is placed on the number of rows returned (default). Setting *value* to a negative number is invalid and is automatically reset to 0.

Remarks

The **MaxRows** property limits the number of rows processed by the remote server. When **MaxRows** is set to a value greater than 0, only '*n'* rows are processed. When executing a query that returns rows, it means that only the first '*n'* rows are returned. When executing an action query, it means that only the first '*n'* rows are updated, inserted or deleted.

This property is useful in situations where limited resources prohibit management of large numbers of result set rows. By setting **MaxRows** to 1 on an action query, you can be assured that no more than one row will be affected by the operation.

Name Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproNameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproNameX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproNameS"}

Returns the name of a **RemoteData** object.

Syntax

object.**Name**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Name** property returns a string expression that represents the name assigned to the object. The following table describes how each object is assigned its name.

Assigning the Name Property for Remote Data Objects

Remote Data Object	Name property is determined by …
rdoEnvironments(0)	**rdoEngine** — Set to "Default_Environment."
rdoEnvironments(1-*n*)	*name* argument of **rdoCreateEnvironment.**
rdoConnection	Data source name (DSN) used for connection.
rdoResultset	First 256 characters of the SQL query.
rdoQuery	*name* argument in **CreateQuery** method or set directly for stand-alone **rdoQuery** objects.
rdoTable	Database table name once the **rdoTables** collection is populated.
rdoParameter	"Param*n*" where "*n*" is the ordinal number.
rdoColumn	Database column name.
rdoError	Not applicable. **rdoErrors** collection members can only be referenced by their ordinal number.

Remarks

rdoTable and **rdoQuery** objects can't share the same name. In other words, you cannot create two **rdoQuery** objects that have the same name.

Use the **Name** property to reference members of a collection in code, but in most cases, it is easier to simply use the ordinal number. Generally, you can use the **Name** property to map database table and column names.

Name Property Example (RDO)

The following example illustrates use of the Name property to expose the names of all tables associated with a chosen database, the names of each column for the selected table, and specific type information about a selected column. This application uses three **Listbox** controls and a **Command** button control.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As rdoConnection
Dim rs As rdoResultset
Dim tb As rdoTable
Dim cl As rdoColumn
Dim er As rdoError

Private Sub Command1_Click()
Set en = rdoEngine.rdoEnvironments(0)

Set cn = en.OpenConnection(dsName:="WorkDB", _
    prompt:=rdDriverNoPrompt, _
    Connect:="Uid=;pwd=;database=Pubs")

For Each tb In cn.rdoTables
    List1.AddItem tb.Name
Next
List1.ListIndex = 1
End Sub

Private Sub List1_Click()
List3.Enabled = False
List2.Clear
For Each cl In cn.rdoTables((List1)).rdoColumns
    List2.AddItem cl.Name
Next
List3.Enabled = True
End Sub

Private Sub List2_Click()
List3.Clear
With cn.rdoTables((List1)).rdoColumns((List2))
    List3.AddItem "Source Column:" & .SourceColumn
    List3.AddItem "Source Table:" & .SourceTable
    List3.AddItem "Type:" & .Type
    List3.AddItem "Size:" & .Size
    List3.AddItem "Ordinal Position:"  _
   & .OrdinalPosition
    List3.AddItem "Allow Zero Length ?" _
    & .AllowZeroLength
    List3.AddItem "Required:" & .Required
    List3.AddItem "Chunk Required?:" & .ChunkRequired
    List3.AddItem "Updatable?:" & .Updatable
End With
End Sub
```

Number Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproNumberC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproNumberX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproNumberA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproNumberS"}

Returns a numeric value specifying a native error.

Syntax

object.**Number**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The return value is a Long integer representing an error number.

Remarks

Use the **Number** property to determine the nature of an error that occurred on the remote server or in the ODBC interface with the data source. The value of the property corresponds to a unique number that corresponds to an error condition generated by a stored procedure, a syntax or other procedural error, a permissions or rule violation or some other type of error. This native number can also be generated by a remote procedure executing a statement such as SQL Server's RAISERROR statement.

Note The SQL Server error severity level is *not* returned by the ODBC driver, and is therefore unavailable.

Error Description and Number Properties Example

The following code opens a read-only ODBC cursor connection against the SQL Server "SEQUEL" and includes a simple error handler that displays the error description and number.

```
Sub MakeConnection()
Dim rdoCn As New rdoConnection
On Error GoTo CnEh
With rdoCn
    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=SEQUEL;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .CursorDriver = rdUseODBC
    .EstablishConnection rdDriverNoPrompt, True
End With
AbandonCn:
Exit Sub

CnEh:
Dim er As rdoError
Dim msg as string
    Msg = "An error occured " _
    & "while opening the connection:" _
    & Err & " - " & Error & VbCr
    For Each er In rdoErrors
        Msg = Msg & er.Description  _
         & ":" & er.Number & VbCr
    Next er
    Resume AbandonCn
End Sub
```

Options Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproOptionsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproOptionsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproOptionsS"}

Returns or sets a value that specifies one or more operational characteristics of the **RemoteData** control.

Syntax

object.**Options** [= *value*]

The **Options** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or **Integer** as described in Settings.

Settings

Use the following values to set the **Options** property for the **RemoteData** control:

Constant	Value	Description
rdAsyncEnable	32	Execute the query asynchronously.
rdExecDirect	64	Use the ODBC **SQLExecDirect** API function to execute query.

Remarks

This property corresponds to the *options* argument in the **OpenResultset** and **Execute** methods. If you change the **Options** property at run time, you must use the **Refresh** method for the change to have any effect.

Enable Asynchronous Operations

Asynchronous operations permit **RemoteData** objects to work in the background on operations like creating result sets or executing procedures while your foreground code continues to work.

Whenever you use the **OpenResultset**, **Execute**, **Move** or **MoreResults** methods with the **rdAsyncEnable** option, control returns immediately to your application — before the operation is completed by RDO. If required, RDO periodically checks the data source to see if the operation is complete. You can adjust the frequency of this polling by setting the **AsyncCheckInterval** property. To see if your operation has completed, check the **StillExecuting** property which remains **True** until RDO completes the operation. To cancel the operation, use the **rdoResultset** object's **Cancel** method. In addition, when queries are complete, RDO fires the QueryComplete event to indicate that the **rdoResultset** is ready to access.

Enable Use of SQLExecDirect

If you use the **rdExecDirect** option, RDO uses the **SQLExecDirect** ODBC API function to execute the query. In this case, no temporary stored procedure is created to execute the query. This option can save time if you don't expect to execute the query more than a few times in the course of your application. In addition, when working with queries that should not be run as stored procedures but

executed directly, this option is mandatory. For example, in queries that create temporary tables for use by subsequent queries, you must use the **rdExecDirect** option.

OrdinalPosition Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproOrdinalPositionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproOrdinalPositionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproOrdinalPositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproOrdinalPositionS"}

Returns the relative position of an **rdoColumn** object within the **rdoColumns** collection.

Syntax

object.**OrdinalPosition**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **OrdinalPosition** property return value is an Integer expression as described in Remarks.

Remarks

Each of these data columns in an **rdoResultset** is returned in the order specified by the query and are numbered starting at one. Because of this numbering scheme, the first column of an **rdoResultset** (`rdoResultset(0)`) has an **OrdinalPosition** of 1.

OrdinalPosition Property Example

This example illustrates how columns are numbered with the **OrdinalPosition** property.

```
Private Sub ShowOrdinals()
Dim rs As rdoResultset
Dim cn As New rdoConnection
Dim cl As rdoColumn
Dim er As rdoError

On Error GoTo EH
With cn
    .Connect = "uid=;pwd=;server=SEQUEL;" _
    & "driver={SQL Server};database=pubs;" _
    & "DSN='';"
    .CursorDriver = rdUseNone
    .EstablishConnection rdDriverNoPrompt
    Set rs = cn.OpenResultset( _
      "select * from titles where Price < 50")
    For Each cl In rs.rdoColumns
        Debug.Print cl.OrdinalPosition,  _
         cl.Name, cl.Value
    Next
End With
Exit Sub
EH:
    For Each er In rdoErrors
        Debug.Print er.Description
    Next er
    Resume Next
End Sub
```

Password Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproPasswordC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproPasswordA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproPasswordS"}

Represents the password used during creation of an **rdoEnvironment** object.

Syntax

object.**Password**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **rdoDefaultPassword** property of the **rdoEngine** object is used as a default if no password is provided. The initial default password is "".

This property setting is write-only — it may only be provided in code, it cannot be read back from the **Password** property.

The password is set:

- When the **rdoEnvironment** is created automatically by the **RemoteData** control.
- By the first reference to a **RemoteData** object.
- When the **rdoCreateEnvironment** method is executed.
- In the connect string via the **Connect** property or the **Connect** argument of the **OpenConnection** method.

PercentPosition Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproPercentPositionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproPercentPositionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproPercentPositionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproPercentPositionS"}

Returns or sets a value that indicates or changes the approximate location of the current row in the **rdoResultset** object based on a percentage of the rows in the **rdoResultset**.

Syntax

object.**PercentPosition** [= *value*]

The **PercentPosition** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A number between 0.0 and 100.00. (Data type is Single)

Remarks

To indicate or change the approximate position of the current row in an **rdoResultset**, you can check or set the **PercentPosition** property. Before you set or check the **PercentPosition** property, populate the **rdoResultset** by moving to the last row. If you use the **PercentPosition** property *before* fully populating the **rdoResultset**, the amount of movement is relative to the number of rows accessed — as indicated by the **RowCount** property. You can move to the last row and populate the **rdoResultset** using the **MoveLast** method.

Note Using the **PercentPosition** property to move the current row to a specific row in an **rdoResultset** isn't recommended — the **Bookmark** property or **AbsolutePosition** property is better suited for this task.

Once you set the **PercentPosition** property to a value, the row at the approximate position corresponding to that value becomes current, and the **PercentPosition** property is reset to a value that reflects the approximate position of the current row. For example, if your **rdoResultset** contains only five rows, and you set its **PercentPosition** value to 77, the value returned from the **PercentPosition** property might be 80, not 77.

You can use the **PercentPosition** property with a scroll bar on a **Form** or **TextBox** to indicate the location of the current row in an **rdoResultset**.

The **PercentPosition** property is not supported by all cursor types and driver combinations. For example, this property applies only to keyset-type and dynamic-type **rdoResultset** objects. If the setting is not supported, the **PercentPosition** property returns 50. If the position cannot be set, no movement occurs.

PercentPosition Property Example

This example illustrates use of the **PercentPosition** property. In this example a list of publishers is generated and when one of these is chosen, a list of associated titles is displayed in a **DBGrid** control. When the scroll bar associated with the grid is manipulated, the relative location of the selected row is determined by examining the **PercentPosition** property and displayed. See the **AbsolutePosition** property example for further details on this example.

```
Dim rs As rdoResultset

Private Sub Form_Load()
Dim Li As Integer
'
'   Fill Sections list combo box.
'
Set en = rdoEnvironments(0)
Set cn = en.OpenConnection(dsName:="", _
    Prompt:=rdDriverNoPrompt, _
    Connect:="uid=;pwd=;driver={SQL Server};" _
        & "server=BETAV486;database=pubs;")

MsRdc1.Connect = cn.Connect

Set rs = cn.OpenResultset _
    ("Select distinct Pub_Name, Pub_ID from Publishers", rdOpenStatic,
rdConcurReadOnly)
Do Until rs.EOF
    If rs(0) = Null Then
    Else
        PubList.AddItem " " & rs!Pub_ID & ":" & rs!Pub_Name
    End If
    rs.MoveNext
Loop
PubList.ListIndex = 1
rs.Close

Publist_Click


End Sub

Private Sub MoveCRow_Change()
MoveCRow_Scroll
End Sub

Private Sub MoveCRow_Scroll()
PercentPoint = MoveCRow.Value & "%"
MsRdc1.Resultset.PercentPosition = MoveCRow.Value
End Sub

Private Sub Publist_Click()
SetSQL
If MsRdc1.Resultset.EOF Then
    MoveCRow.Enabled = False
Else
    MoveCRow.Enabled = True
```

```
    MsRdc1.Resultset.MoveFirst
End If
End Sub

Sub SetSQL()
Dim PubWanted As String
PubWanted = Trim(Left(PubList, InStr(PubList, ":") - 1))
Screen.MousePointer = vbHourglass

MsRdc1.SQL = "select * from Titles" _
    & " where Pub_ID = '" _
    & PubWanted & "'" _
    & " order by Title"
MsRdc1.Refresh
Screen.MousePointer = vbDefault
End Sub
```

Prompt Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproPromptC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproPromptX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproPromptA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproPromptS"}

Returns or sets a value that specifies if the ODBC driver manager should prompt for missing connect string arguments.

Syntax

object.**Prompt** [= *value*]

The **Prompt** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or **Integer** as described in Settings.

Settings

The settings for the **Prompt** property are:

Constant	Value	Description
rdDriverPrompt	0	The driver manager displays the ODBC Data Sources dialog box. The connection string used to establish the connection is constructed from the data source name (DSN) selected and completed by the user via the dialog boxes. Or, if no DSN is chosen and the **DataSourceName** property is empty, the default DSN is used.
rdDriverNoPrompt	1	The driver manager uses the connection string provided in *connect*. If sufficient information is not provided, the **OpenConnection** method returns a trappable error.
rdDriverComplete	2	If the connection string provided includes the DSN keyword, the driver manager uses the string as provided in *connect*, otherwise it behaves as it does when **rdDriverPrompt** is specified.
rdDriverCompleteRequired	3	(Default) Behaves like **rdDriverComplete** except the driver disables the controls for any information not required to complete the connection.

Remarks

When RDO opens a connection based on the parameters of the **RemoteData** control, the **Connect** property is expected to contain sufficient information to establish the connection. If information like the data source name, user name, or password are not provided, the ODBC driver manager exposes one or more dialog boxes to gather this information from the user. If you do not want these dialog boxes to

appear, set the **Prompt** property accordingly to disable this feature.

The constants shown above are also used to set the ODBC prompt behavior for the **EstablishConnection** method.

QueryTimeout Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproQueryTimeoutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproQueryTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproQueryTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproQueryTimeoutS"}

Returns or sets a value that specifies the number of seconds the ODBC driver manager waits before a timeout error occurs when a query is executed.

Syntax

object.**QueryTimeout** [= *value*]

The **QueryTimeout** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the number of seconds the driver manager waits before timing out and returning an error.

Remarks

The default **QueryTimeout** property setting is 30 seconds. When you're accessing an ODBC data source using the **OpenResultset** or **Execute** methods, there may be delays due to network traffic or heavy use of the remote server – perhaps caused by your query. Rather than waiting indefinitely, use the **QueryTimeout** property to determine how long your application should wait before the QueryTimeout event is fired and your application trips a trappable error. At this point you have the option to continue waiting for another 'n' seconds as determined by the **QueryTimeout** property, or cancel the query in progress by using the **Cancel** argument in the QueryTimeout event procedure.

Setting this property to 0 disables the timer so your query will run indefinitely. Setting **QueryTimeout** to 0 is *not* recommended for synchronous operations as your application can be blocked for the entire duration of the query.

When used with an **rdoConnection** object, the **QueryTimeout** property specifies a global value for all queries associated with the data source.

When you use an **rdoQuery** object, the **rdoConnection** object's **QueryTimeout** property is used as a default value unless you specify a new value in the **rdoQuery** object's **QueryTimeout** property.

When working with asynchronous queries, the **StillExecuting** property remains **True** until the query completes, or the query timeout period is exhausted.

If the specified timeout exceeds the maximum timeout permitted by the data source, or is smaller than the minimum timeout, the driver substitutes that value and the following error is logged to the rdoErrors collection: SQLState 01S02: "Option value changed."

QueryTimeout Property, QueryTimeout Event Example

The following example sets up the query event handlers to deal with query timeout contingencies. Notice that the QueryTimeout event procedure displays a message box that permits the user to decide if they want to wait for an additional timeout period for the query. The ShowRows procedure simply dumps the rows returned.

```
Option Explicit
Dim en As rdoEnvironment
Dim cn As New rdoConnection
Dim rs As rdoResultset
Dim SQL As String
Dim col As rdoColumn
Dim er As rdoError
Public WithEvents Qd As rdoQuery

Private Sub cn_QueryTimeout( _
   ByVal Query As RDO.rdoQuery, Cancel As Boolean)
Dim ans As Integer
ans = MsgBox("Query Timed out... Press Retry to continue waiting", _
    vbRetryCancel + vbCritical, "Query Took Too Long")
If ans = vbRetry Then
    Cancel = False
Else
    Cancel = True
End If
End Sub

Private Sub RunQuery_Click()

On Error GoTo RunQueryEH

    Qd(0) = Param1
    Qd.QueryTimeout = 5
    Set rs = Qd.OpenResultset(rdOpenKeyset, _
       rdConcurReadOnly)

    If rs Is Nothing Then Else ShowRows

Exit Sub

RunQueryEH:
Debug.Print Err, Error$
    For Each er In rdoErrors
        Debug.Print er.Description, er.Number
    Next
    rdoErrors.Clear
    Resume Next

End Sub


Private Sub Form_Load()
Set en = rdoEngine.rdoEnvironments(0)
With cn
    .Connect = "uid=;pwd=;database=workdb;dsn=WorkDB;"
    .CursorDriver = rdUseClientBatch
```

```
    .EstablishConnection Prompt:=rdDriverNoPrompt
End With

Set Qd = cn.CreateQuery("LongQuery", "")
With Qd
    .SQL = "{call VeryLongStoredProcedure (?,?)}"
    .rdoParameters(1).Direction = rdParamOutput
    .rdoParameters(0).Type = rdTypeVARCHAR
End With

End Sub
```

BatchCollisionCount Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBatchCollisionCountC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBatchCollisionCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBatchCollisionCountS"}

Returns a value that specifies the number of rows that did not complete the last batch-mode update.

Syntax

object.**BatchCollisionCount**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **BatchCollisionCount** property return value is a Long expression that specifies the number of failing rows or 0 if all rows were processed.

Remarks

This property indicates how many rows encountered collisions or otherwise failed to update during the last batch update attempt. The value of this property corresponds to the number of bookmarks in the **BatchCollisionRows** array.

By setting the working **rdoResultset** object's **Bookmark** property to bookmark values in the **BatchCollisionRows** array, you can position to each row that failed to complete the most recent **BatchUpdate** operation.

After the collision rows are corrected, the **BatchUpdate** method can be called again. At this point RDO attempts another batch update, and the **BatchCollisionRows** property again reflects the set of rows that failed the second attempt. Any rows that succeeded in the previous attempt are not sent in the current attempt, as they now have a **Status** of **rdRowUnmodified**. This process can continue as long as collisions occur, or until you abandon the updates and close the result set.

BatchCollisionRows Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBatchCollisionRowsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBatchCollisionRowsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBatchCollisionRowsS"}

Returns an array of bookmarks indicating the rows that generated collisions in the last batch update operation.

Syntax

object.**BatchCollisionRows**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **BatchCollisionRows** property return value is a **Variant**(string) expression as described in Remarks.

Remarks

This property contains an array of bookmarks to rows that encountered a collision during the last invocation of the **BatchUpdate** method. The number of elements in the array is indicated by the **BatchCollisionCount** property.

By setting the working **rdoResultset** object's **Bookmark** property to bookmark values in the **BatchCollisionRows** array, you can position to each row that failed to complete the most recent **BatchUpdate** operation.

After the collision rows are corrected, the **BatchUpdate** method can be called again. At this point RDO attempts another batch update, and the **BatchCollisionRows** property again reflects the set of rows that failed the second attempt. Any rows that succeeded in the previous attempt are not sent in the current attempt, as they now have a **Status** of **rdRowUnmodified**. This process can continue as long as collisions occur, or until you abandon the updates and close the result set.

This array is re-created each time the **BatchUpdate** method executes.

BatchConflictValue Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBatchConflictValueC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBatchConflictValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBatchConflictValueS"}

Returns a value currently in the database that is newer than the **Value** property as determined by an optimistic batch update conflict.

Syntax

object.**BatchConflictValue**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **BatchConflictValue** property return value is a **Variant**(String) expression as described in Remarks.

Remarks

This property contains the value of the column that is currently in the database on the server. During an optimistic batch update, a collision may occur where a second client modified the same column and row in between the time the first client fetched the data and the update attempt. When this happens, the value that the second client set will be accessible through this property.

BatchSize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproBatchSizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproBatchSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproBatchSizeS"}

Returns or sets a value that specifies the number of statements sent back to the server in each batch.

Syntax

object.**BatchSize** [= *value*]

The **BatchSize** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the number of statements sent back to the server in each batch. The default value is 15.

Remarks

This property determines the batch size used when sending statements to the server during an optimistic batch update. The value of the property determines the number of statements sent to the server in one command buffer. By default, 15 statements are sent to the server in each batch. This property can be changed at any time. If a DBMS doesn't support statement batching, you can set this property to 1, causing each statement to be sent separately.

CursorType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproCursorTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproCursorTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproCursorTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproCursorTypeS"}

Returns or sets a value that specifies the default type of cursor to use when opening a result set from the specified query.

Syntax

object.**CursorType** [= *value*]

The **CursorType** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the type of cursor as described by one of the following constants:

Constant	Value	rdoResultset type
rdOpenForwardOnly	0	(Default) Fixed set, non-scrolling**.**
rdOpenKeyset	1	Updatable, fixed set, scrollable query result set cursor.
rdOpenDynamic	2	Updatable, dynamic set, scrollable query result set cursor.
rdOpenStatic	3	Read-only, fixed set.

Remarks

Determines the cursor type to use when opening an **rdoResultset** object from this query.

When creating a stand-alone **rdoQuery** object whose query is to be used as a method, you should set the **CursorType** before the query is executed because there is no option to do so when the query is executed.

The value of the **CursorType** property is used as the **Type** argument of the **OpenResultset** method.

Not all cursor libraries support all types of cursors. For example, the ODBC client-side driver can only support **rdOpenStatic** and **rdOpenForwardOnly** cursor types, while the SQL Server server-side driver supports all four types. Generally, most drivers support forward-only and static cursors.

CursorType Property Example

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim col As rdoColumn

Private Sub Form_Load()

cn.CursorDriver = rdUseClientBatch
cn.Connect = "uid=;pwd=;server=sequel;" _
   & "driver={SQL Server};database=pubs;dsn='';"
cn.EstablishConnection

'
'  Setup the query
'
With qy
    .Name = "ShowAuthor"
    .SQL = "Select * from Authors " _
      & "where Au_LName = ? "
    .LockType = rdConcurReadOnly
    .CursorType = rdOpenForwardOnly
    .RowsetSize = 1
    Set .ActiveConnection = cn
End With

'
'  Execute the Query by Name
'  Pass in a parameter to the query
'
cn.ShowAuthor "White"
'
'  Process the resulting rows
'

If cn.LastQueryResults is Nothing then
Else
   Set rs = cn.LastQueryResult
   For Each col In rs.rdoColumns
      Print col.Name,
   Next
   Print String(80, "-")

   Do Until rs.EOF
      For Each col In rs.rdoColumns
         Print col,
      Next
      Print
      rs.MoveNext
   Loop
End if

End Sub
```

KeyColumn Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproKeyColumnC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproKeyColumnA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproKeyColumnS"}

Returns or sets a value that specifies if this column is part of the primary key.

Syntax

object.**KeyColumn** [= *value*]

The **KeyColumn** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression as described in Settings.

Settings

The **KeyColumn** property has these settings:

Setting	Description
True	If the column is part of the primary key.
False	(Default) If the column is not part of the primary key.

Remarks

This property indicates if the column is part of the primary key for the result set. This property will be read/write when using the client batch cursor library (**CursorDriver** property set to **rdUseClientBatch**) and generates a trappable error when accessed using server-side cursors or ODBC cursor library.

When using the client batch cursor library, you can set this property to indicate which columns make up the primary key of the result set. This assists the cursor library when it builds the WHERE clauses for the update or delete/insert statements during an optimistic batch update.

LastQueryResults Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproLastQueryResultsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproLastQueryResultsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproLastQueryResultsS"}

Returns a reference to the **rdoResultset** object generated by the last query – if any.

Syntax

object.**LastQueryResults**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **LastQueryResults** property return value is an object expression that specifies a valid **rdoResultset** or Nothing if there is no result set available.

Remarks

This new property will contain a reference to the **rdoResultset** object generated by the last query executed on this connection, if any. The necessity of this property comes from the Queries as Methods feature, allowing the developer to call their queries and stored procedures as methods of the parent connection object. Since stored procedures can pass back a return value as well as resultsets, the developer needs this property in order to get a reference to the result set created during the last query as method call. The developer would use this property as shown below:

```
Dim RetCode As Long
Dim rs as rdoResultset
RetCode = MyConnection.MyQuery(x,y,z)
Set rs = MyConnection.LastQueryResults
```

This property is set back to “Nothing” on the next query execution on the connection. This property will be set for any query executed on the connection, even if the developer used the OpenResultset or Execute methods instead of calling the query as a method of the connection object.

Prepared Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproPreparedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproPreparedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproPreparedS"}

Returns or sets a value that determines if the query should be prepared using the **SQLPrepare** or **SQLExecDirect** ODBC API function.

Syntax

object.**Prepared** [= *value*]

The **Prepared** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Boolean** expression as described in Settings.

Settings

The **Prepared** property has these settings:

Setting	Description
True	The statement is prepared. (Default)
False	The statement is not prepared.

Remarks

By default the **Prepared** property is **True**. However, you can set this property to **False** to prohibit "preparation" of the query. In this case, the query is executed using the **SQLExecDirect** API.

When the ODBC interface submits a query to the remote server, it either submits the query directly to the server, or creates a stored procedure to perform the operation. Creating a stored procedure can slow down the initial operation, but increases performance of all subsequent references to the query. However, some queries cannot be executed in the form of stored procedures. In these cases, you must set the **Prepare** property to **False**.

Status Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproStatusC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproStatusA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproStatusS"}

Returns or sets the status of a row.

This property indicates or sets the status of the current row.

Syntax

object.**Status** [= *value*]

The **Status** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the type of cursor as described in Settings:

Settings

The **Status** property has these settings:

Constant	Value	Prepared **Property Setting**
rdRowUnmodified	0	(Default) The row has not been modified or has been updated successfully.
rdRowModified	1	The row has been modified and not updated in the database.
rdRowNew	2	The row has been inserted with the **AddNew** method, but not yet inserted into the database.
rdRowDeleted	3	The row has been deleted, but not yet deleted in the database.
rdRowDBDeleted	4	The row has been deleted locally *and* in the database.

Remarks

The value of this property indicates if and how this row will be involved in the next optimistic batch update.

When you use the optimistic batch update cursor library and need to specify which rows are to be updated in the next batch operation, you set the **rdoResultset** object's **Status** property. For example, suppose you are working with an unbound Grid control filled with rows from a query. The user selects one of the rows and you detect that a change has been made in the row. At this point you can mark this row for updating by setting the **Status** property to **rdRowModified**. Similarly, if a row is added or deleted, you can use the appropriate **Status** property setting to so indicate. When you use the **BatchUpdate** method, RDO will submit an appropriate operation to the remote server for each row based on its **Status** property.

Once the **BatchUpdate** operation is complete, you can examine the **Status** property of each row to determine if the update is successful. If the **Status** value does not return rdRowUnmodified after the **BatchUpdate**, the operation to update the row could not be completed. In this case you should check the **rdoErrors** collection and the **BatchCollisionRows** property for rows.

UpdateCriteria Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproUpdateCriteriaC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproUpdateCriteriaA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproUpdateCriteriaS"}

Returns or sets a value that specifies how the WHERE clause is constructed for each row during an optimistic batch update operation.

Syntax

object.**UpdateCriteria** [= *value*]

The **UpdateCriteria** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the type of cursor as described in Settings

Settings

The **UpdateCriteria** property has these settings:

Constant	Value	rdoResultset type
rdCriteriaKey	0	(Default). Uses just the key column(s) in the where clause.
rdCriteriaAllCols	1	Uses the key column(s) and all updated columns in the where clause.
rdCriteriaUpdCols	2	Uses the key column(s) and all the columns in the where clause.
rdCriteriaTimeStamp	3	Uses just the timestamp column if available (will generate a runtime error if no timestamp column is in the result set).

Remarks

When a batch mode operation is executed, RDO and the ClientBatch cursor library create a series of UPDATE statements to make the needed changes. An SQL WHERE clause is created for each update to isolate the rows that are marked as changed (by the **Status** property). Because some remote servers use triggers or other ways to enforce referential integrity, is it often important to limit the columns being updated to just those affected by the change. This way, only the absolute minimum amount of trigger code is executed. As a result, the update operation is executed more quickly, and with fewer potential errors.

You should set the **UpdateCriteria** property to **rdCriteriaKey** when BLOB columns are included in the result set.

Setting this property to a value other than the ones listed here results in a runtime error.

UpdateOperation Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproUpdateOperationC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproUpdateOperationA"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproUpdateOperationS"}

Returns or sets a value that specifies if the optimistic batch update should use an Update statement or a Delete followed by an Insert.

Syntax

object.**UpdateOperation** [= *value*]

The **UpdateOperation** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A **Long** integer representing the type of cursor as described in Settings:

Settings

The **UpdateOperation** property has these settings:

Constant	Value	UpdateOperation type
rdOperationUpdate	0	(Default) Uses an Update statement for each modified row.
rdOperationDelIns	1	Uses a pair of Delete and Insert statements for each modified row.

Remarks

Setting the **UpdateOperation** property to a value other than the ones listed here results in a runtime error.

This property determines whether the optimistic batch update cursor library uses an update statement, or a pair of delete and insert statements when sending modifications back to the database server. In the latter case, two separate operations are required to update the row. In some cases, especially where the remote system implements Delete, Insert and Update triggers, choosing the correct **UpdateOperation** property can significantly impact performance.

Newly added rows will always generate Insert statements and deleted rows will always generate Delete statements, so this property only applies to how the cursor library updates modified rows.

OriginalValue Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproOriginalValueC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproOriginalValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproOriginalValueS"}

Returns the value of the column as first fetched from the database.

Syntax

object.**OriginalValue**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **OriginalValue** property return value is a **Variant** value whose datatype is determined by the datatype of the rdoColumn specified.

Remarks

When working with optimistic batch update operations, you might need to resolve update conflicts by comparing the column values as originally returned by RDO with the value as supplied by the user. The **OriginalValue** property provides this value as first fetched from the database.

rdoDefaultCursorDriver Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprordoDefaultcursordriverC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprordoDefaultcursordriverA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprordoDefaultCursorDriverS"}

Returns or sets the cursor library used by the ODBC driver manager.

Syntax

object.**rdoDefaultCursorDriver** [= *value*]

The **rdoDefaultCursorDriver** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An Integer constant or value that specifies a type of ODBC cursor as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
rdUseIfNeeded	0	(Default)RDO chooses the style of cursors most appropriate for the driver. Server-side cursors are used if they are available.
rdUseODBC	1	RDO uses the ODBC cursor library. This option gives better performance for small result sets, but degrades quickly for larger result sets.
rdUseServer	2	RDO uses server-side cursors. For most large operations this gives better performance, but might cause more network traffic.
rdUseClientBatch	3	RDO uses the optimistic batch cursor library as required by all batch mode operations and dissociate **rdoResultset** objects.
rdUseNone	4	RDO does not create a scrollable cursor. Basically, this is a forward-only, read-only resultset with a **RowsetSize** set to 1. This type of resultset performs faster than those that require creation of a cursor.

Remarks

When server-side cursors are used, the database engine uses its own resources to store keyset values. Data values are still transmitted over the network as with client-side cursors, but the impact on local workstation memory and disk space is reduced.

For SQL Server, server-side cursors are not used if the cursor is read-only and forward-only.

rdoDefaultUser, rdoDefaultPassword Properties

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprordoDefaultrdoUserC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprordoDefaultrdoUserA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprordoDefaultrdoUserS"}

- **rdoDefaultUser** — returns or sets the default user name assigned to any new **rdoEnvironment**.
- **rdoDefaultPassword** — returns or sets the default password assigned to any new **rdoEnvironment**.

Syntax

object.**rdoDefaultUser** [= *value*]
object.**rdoDefaultPassword** [= *value*]

The syntax for the **rdoDefaultUser** and **rdoDefaultPassword** properties have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that specifies either a user name or password.

Remarks

Unless other values are supplied in the **rdoCreateEnvironment** method, the **rdoDefaultUser** and **rdoDefaultPassword** properties determine the user name and password used when the **rdoEnvironment** object is created. These properties can also return the name used when an **rdoEnvironment** is created.

By default, the *value* for **rdoDefaultUser** and **rdoDefaultPassword** is **""** (a zero-length string).

rdoDefaultErrorThreshold Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprordoDefaultErrorThresholdC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprordoDefaultErrorThresholdA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprordoDefaultErrorThresholdS"}

Returns or sets a value that indicates the default value for the **ErrorThreshold** property for rdoQuery objects.

Note This property is provided for backward compatibility with RDO version 1.0 code. It should be replaced with code that implements the **rdoEngine** object's InfoMessage event which provides equivalent functionality.

Remarks

In version 4.x of Microsoft SQL Server, it is not possible to set the severity of errors using the RAISERROR statement. As a result, the ErrorThreshold property was needed to permit your code to filter those messages beyond a threshold of severity.

Version 6.x of Microsoft SQL Server now supports the inclusion of a severity level in the RAISERROR statement so it is no longer necessary to use the ErrorThreshold property.

All errors that are returned with a severity of less than 10 are trapped by the ODBC layers and set the SQL_SUCCESS_WITH_INFO result code. This causes RDO to raise the InfoMessage event but not stop query processing.

rdoDefaultLoginTimeout Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprordoDefaultLoginTimeoutC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprordoDefaultLoginTimeoutA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprordoDefaultLoginTimeoutS"}

Returns or sets a default value that determines the number of seconds the ODBC driver waits before abandoning an attempt to connect to a data source.

Syntax

object.**rdoDefaultLoginTimeout** [= *value*]

The **rdoDefaultLoginTimeout** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Long expression that specifies the number of seconds as described in Settings.

Settings

The setting for *value* is the number of seconds to wait for a login request to complete before returning a trappable error or firing a QueryTimeout event. A setting of 0 indicates the timeout is disabled, and a connection attempt will wait indefinitely — or until the connection is complete.

Remarks

The **rdoDefaultLoginTimeout** property is used as an application-wide default unless the **LoginTimeout** property of the **rdoEnvironment** object is used to override this value.

Login requests are made when the RemoteData control creates an **rdoConnection object**, or when you use the **OpenConnection** or **EstablishConnection** methods of the **rdoEnvironment** object. The maximum value is dependent on the data source driver. The ODBC driver determines the maximum permissible LoginTimeout value — any attempt to set a value higher than this value is reset to this driver-dependent maximum value.

The default timeout value, if not specified, is 15 seconds.

When the timeout period is exhausted, the ConnectionTimeout event on the parent **rdoEnvironment** object fires.

Note When you use Data Access Objects (DAO), the LOGINTIMEOUT argument is used in the **Connect** property, this is not a valid argument for ODBC connect strings. Use the **rdoDefaultLoginTimeout** property instead.

rdoLocaleID Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRdoLocaleIDC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRdoLocaleIDA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRdoLocaleIDS"}

Returns or sets a value indicating the locale of the RDO library.

Syntax

object.**rdoLocaleID** [= *value*]

The **rdoLocaleID** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that specifies a locale as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
rdLocaleSystem	0	System
rdLocaleEnglish	1	English
rdLocaleFrench	2	French
rdLocaleGerman	3	German
rdLocaleItalian	4	Italian
rdLocaleJapanese	5	Japanese
rdLocaleSpanish	6	Spanish
rdLocaleChinese	7	Chinese
rdLocaleSimplifiedChinese	8	Simplified Chinese
rdLocaleKorean	9	Korean

Remarks

The locale determines which language is used when generating RDO error messages. The **rdoLocaleID** defaults to the Windows system locale when the **rdoEngine** is initialized.

You can override the current locale at any time by setting the **rdoLocaleID** to any of the supported values. If you use an unsupported value, a trappable error occurs.

When the **rdoLocaleID** property is set or changed, RDO loads the appropriate language dynamic-link library (DLL) to show error messages in the correct language.

If the specified language DLL is not present on the user's machine, RDO is set to **rdLocaleEnglish**, which does not require a separate DLL. When this happens, an informational message is placed in the **rdoErrors** collection indicating that RDO was unable to load the resource DLL for the specified locale.

When you distribute your application, be sure to include the appropriate language DLL.

rdoVersion Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdprordoVersionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdprordoVersionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdprordoVersionS"}

Returns a value that indicates the version of the RDO library associated with the object.

Syntax

object.**rdoVersion**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **rdoVersion** property return value is a 5-character string expression.

Remarks

This property identifies the version of the database engine that created the connection. The version is in the form #.#.####, where the first two digits are the major version number and the last two digits are the minor version. For example, RDO version 2.0 returns 2.0.0000.

Resultset Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproResultSetC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproResultsetX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproResultSetA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproResultsetS"}

Returns or sets an **rdoResultset** object defined by a RemoteData control or as returned by the **OpenResultset** method.

Syntax

Set *object*.**Resultset** [= *value*]

The **Resultset** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An object expression that evaluates to an **rdoResultset** object as described in Settings.

Settings

The setting for *value* is an **rdoResultset** object.

Remarks

The **RemoteData** control is automatically initialized when your application starts. If the **SQL** property is valid, or if you set the **SQL** property at run time and use the **Refresh** method, the **RemoteData** control attempts to create a new **rdoResultset** object. This **rdoResultset** is accessible through the **RemoteData** control's **Resultset** property.

If you create an **rdoResultset** object using either RDO code or another **RemoteData** control, you can set the **Resultset** property of the **RemoteData** control to this new **rdoResultset**. Any existing **rdoResultset** in the **RemoteData** control, and the **rdoConnection** object associated with it, are released when a new **rdoResultset** is assigned to the **Resultset** property.

You can also create an **rdoResultset** object using the **OpenResultset** method and setting the **Resultset** property to the resulting **rdoResultset** object. However, the bound controls using the **RemoteData** control must correctly specify the columns of the new **rdoResultset**. To do so, make sure the **DataField** properties of the bound controls connected to the **RemoteData** control are set to match the new **rdoResultset** object's column names.

Type of Result Set

You can also determine the type of **rdoResultset** cursor by examining or setting the **RemoteData** control's **ResultsetType** property. If you don't request a specific type when using the **RemoteData** control, a keyset-type **rdoResultset** is created. You can determine the type of **rdoResultset** at run time by examining the **rdoResultset** object's **Type** property or the **RemoteData** control's **ResultsetType** property.

The **RemoteData** control can create either keyset- or static-type **rdoResultset** objects when accessing SQL Server 6. However, if the ODBC driver does not support keyset cursors, they cannot be created — all drivers support static cursors. A trappable error is triggered if you set the **RemoteData** control's **Resultset** property to an unsupported type of **rdoResultset**.

Note When the **Resultset** property is set, the **RemoteData** control doesn't close the current **rdoResultset** or **rdoConnection,** but it does release it. If there are no other users, the **rdoConnection** is closed automatically. You may want to consider closing the **rdoResultset** and **rdoConnection** associated with the **RemoteData** control before setting the **Resultset** property.

All **rdoResultset** objects created by the **RemoteData** control are built in **rdoEnvironments**(0). If you need to use the **RemoteData** control to manipulate a database in another **rdoEnvironment**, use the technique demonstrated in the example to open the **rdoConnection** in the desired **rdoEnvironment**, create a new **rdoResultset**, and set the **RemoteData** control's **Resultset** property to this new **rdoResultset**.

Resultset Property Example

The following example shows how to create an **rdoResultset** in code and pass it to an existing **RemoteData** control:

```
Option Explicit
Dim qy As rdoQuery
Dim rs As rdoResultset
Dim cn As rdoConnection

Private Sub Form_Load()
Dim SQL As String
Set cn = MSRDC1.Connection

SQL = "{ call ChooseAuthor (?) }"
Set qy = cn.CreateQuery("GetAuthor", SQL)
End Sub


Private Sub Search_Click()
qy(0) = NameWanted.Text
Set MSRDC1.Resultset = qy.OpenResultset( _
   rdOpenStatic, rdConcurReadOnly)

End Sub
```

The stored procedure executed by the query example is shown below:

```
CREATE PROCEDURE ChooseAuthor (@authorwanted char(20)) as
select t.title from titles t, titleauthor ta, authors a
where t.title_id = ta.title_id
and ta.au_id = a.au_id
and a.au_lname = @authorWanted
```

ResultsetType Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproResultsetTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproResultsetTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproResultsetTypeS"}

Returns or sets a value indicating the type of **rdoResultset** cursor created or to create.

Syntax

object.**ResultsetType** [= *value*]

The **ResultsetType** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or value that specifies a type of **rdoResultset**, as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
rdOpenStatic	3	A static-type **rdoResultset**.
rdOpenKeyset	1	(Default)A keyset-type **rdoResultset**.

Remarks

Not all drivers support all types of cursors. For example, SQL Server 6 supports both static and keyset cursors, but SQL Server 4.2 only supports static cursors. If the ODBC driver does not support keyset cursors, they cannot be created by RDO or the RemoteData control. If the **RemoteData** control can't create the type of **rdoResultset** cursor requested, RDO builds one of the types that can be created and returns the cursor type in the **ResultsetType** property.

If you don't specify a **ResultsetType** before the **RemoteData** control creates the **rdoResultset**, a keyset type **rdoResultset** is created.

If you create an **rdoResultset** and set the **Resultset** property with this new object, the **ResultsetType** property of the **RemoteData** control is set to the **Type** property of the new **rdoResultset**.

ReadOnly Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdProReadOnlyC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdProReadOnlyA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproReadOnlyS"}

Returns or sets a value that determines whether the control's rdoConnection is opened for read-only access.

Syntax

object.**ReadOnly** [= *value*]

The **ReadOnly** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines read/write access, as described in Settings.

Settings

The settings for *value* are:

Setting	Description
True	The **RemoteData** control's **rdoConnection** object is opened with read-only access. Changes to data aren't allowed.
False	(Default) The **RemoteData** control's **rdoConnection** is opened with read/write access to data.

Remarks

Use the **ReadOnly** property with a **RemoteData** control to specify whether data in the underlying **rdoConnection** can be changed. For example, you might create an application that only displays data. Accessing a read-only **rdoConnection** might be faster.

Even if the **ReadOnly** property is **False**, a user might not have write access to a database because the user does not have permission or the type of **rdoResultset** in use does not support updates.

This property corresponds to the **OpenConnection** method's ***readonly*** argument.

Required Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRequiredC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRequiredA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRequiredS"}

Returns a value that indicates whether an **rdoColumn** requires a non-Null value.

Syntax

object.**Required**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The return values for the **Required** property are:

Value	Description
True	A **Null** value isn't allowed.
False	A **Null** value is allowed.

Remarks

For an **rdoColumn** object, you can use the **Required** property along with the **AllowZeroLength** property to determine the validity of the **Value** property setting for that **rdoColumn** object. If **Required** is set to **False**, the column can contain **Null** values as well as values that meet the conditions specified by the **AllowZeroLength** property setting.

Restartable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRestartableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRestartableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRestartableS"}

Returns a value that indicates whether an **rdoResultset** object supports the **Requery** method, which re-executes the query the **rdoResultset** is based on.

Syntax

object.**Restartable**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Restartable** property return values are:

Value	Description
True	The **rdoResultset** object supports the **Requery** method.
False	The **rdoResultset** object doesn't support the **Requery** method.

Remarks

Check the **Restartable** property before using the **Requery** method on an **rdoResultset**. If the object's **Restartable** property is set to **False**, use the **OpenResultset** method on the underlying rdoQuery to re-execute the query.

You can use the **Requery** method to update an **rdoResultset** object's underlying parameter query after the parameter values have been changed.

If the **rdoQuery** does not contain parameters, the **Restartable** property is always **True**.

RowCount Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRowCountC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproRowCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRowCountA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRowCountS"}

Returns the number of rows accessed in an **rdoResultset** object.

Syntax

object.**RowCount**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **RowCount** property return value is a Long integer as discussed in Remarks.

Remarks

Use the **RowCount** property to find out how many rows in an **rdoResultset** object have been accessed, or if the **rdoResultset** returned any rows at all. **RowCount** doesn't indicate how many rows will be returned by an **rdoResultset** query until all rows have been accessed. *After* all rows have been accessed, the **RowCount** property reflects the total number of rows in the **rdoResultset**.

Referencing the **RowCount** property causes RDO to fully-populate the result set — just as if you had executed a **MoveLast** method.

Depending on the driver and data source, the RowCount property might return either -1 to indicate that the number of rows is not available, or 0 to indicate that no rows were returned by the **rdoResultset**. If the driver is capable of returning a row count, the **RowCount** property returns the number of rows in the **rdoResultset**.

Using the **Requery** method on an **rdoResultset** resets the **RowCount** property, just as it does when a query is run for the first time.

RowCount Property Example

This example illustrates use of the **RowCount** property to determine if any rows resulted from the query and how many rows have been processed after each **MoveNext** method is executed and how many rows are in the result set once all rows are processed.

```
Option Explicit
Dim ps As rdoPreparedStatement
Dim rs As rdoResultset
Dim cn As rdoConnection

Private Sub Form_Load()
Dim SQL As String
Set cn = MSRDC1.Connection

SQL = "{call ChooseAuthor (?)}"
Set ps = cn.CreatePreparedStatement("GetAuthor", SQL)
End Sub


Private Sub Search_Click()
ps(0) = NameWanted.Text
Set MSRDC1.Resultset = ps.OpenResultset( _
   rdOpenKeyset, rdConcurReadOnly)
With MSRDC1.Resultset
  If .RowCount = 0 Then
    MsgBox "No titles by that author were found."
    NameWanted.SetFocus
  Else
    Do Until .EOF
      .MoveNext
      TitlesFound = .RowCount      ' How many so far
    Loop
    TitlesFound = .RowCount        ' Total rows found
  End If
  .MoveFirst
End With
End Sub
```

RowsAffected Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRowsAffectedC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproRowsAffectedX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRowsAffectedA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRowsAffectedS"}

Returns the number of rows affected by the most recently invoked **Execute** method.

Syntax

object.**RowsAffected**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **RowsAffected** property return value is a Long value ranging from 0 to the number of rows affected by the most recently invoked **Execute** method on either an rdoConnection or rdoQuery object.

Remarks

RowsAffected contains the number of rows deleted, updated, or inserted when running an action query. When you use the **Execute** method to run an **rdoQuery**, the **RowsAffected** property setting is the number of rows affected. For example, when you execute a query that deletes 50 rows from a table, the **RowsAffected** property returns 50.

This property is especially useful when you need to determine how many rows were affected in an action query with an ambiguous WHERE clause. For example, in a query that deletes all rows where the *Price* column is greater than 10, the **RowsAffected** property would indicate how many rows actually qualified and were deleted.

RowsAffected Property Example

This example illustrates use of the **RowsAffected** property to determine the number of rows changed by an ambiguous update statement. In this case we do not know how many authors live in Salt Lake City, but once the update takes place, we can determine how many rows were updated by checking the **RowsAffected** property of the **rdoQuery** object.

```
Option Explicit
Dim SQL As String
Dim cn as rdoConnection
Dim qd as rdoQuery

Private Sub Form_Load()
Set cn = rdoEnvironments(0).OpenConnection( _
  dsname:="WorkDB", _
  Prompt:=rdDriverNoPrompt, _
  Connect:="uid=;pwd=;database=pubs")

SQL = "Update Authors Set Zip = ?" _
  & " Where City = ?"
Set qd = cn.CreateQuery("FixZip", SQL)
End Sub

Private Sub UpdateButton_Click()
qd(0) = NewZip    ' From a TextBox
qd(1) = CityToFind' From a TextBox
ps.Execute
NumberChanged = qd.RowsAffected
End Sub
```

RowsetSize Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproRowsetsizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproRowsetsizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproRowsetsizeS"}

Returns or sets a value that determines the number of rows in an **rdoResultset** cursor.

Syntax

object.**RowsetSize** [= *value*]

The **RowsetSize** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A value that specifies the size of the rowset as described in Settings. (Data type is a Long expression.)

Settings

The upper limit of the **RowsetSize** is determined by the data source driver. The lower limit for *value* is 1, and the default value is 100.

Remarks

The **RowsetSize** property determines how many rows of the keyset are buffered by the application. RDO uses the **RowsetSize** property to determine how many rows are read into memory with the ODBC **SQLExtendedFetch** function. Tuning the size of **RowsetSize** can affect performance and the amount of memory required to maintain the keyset buffer.

This property must be set before creating an **rdoResultset** object.

Impact on Pessimistic Cursors

In addition, the **RowsetSize** property determines how many rows (and data pages) are locked when using Pessimistic (**rdConcurLock**) concurrency. For example, if you set the **RowsetSize** property to 100 and execute an **rdoQuery** object with the rdConcurLock option, the first 100 rows of the result set and every page associated with each row is locked until the cursor is closed or you move the current row pointer toward the end of the result set. In any case, at least **RowsetSize** rows are locked.

Size Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSizeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSizeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSizeS"}

Returns a value that indicates the maximum size, in bytes, of the underlying data of an **rdoColumn** object that contains text or the fixed size of an **rdoColumn** object that contains text or numeric values.

Syntax

object.**Size**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Return Values

The **Size** property return value is a Long value. The value depends on the **Type** property setting of the **rdoColumn** object, as discussed in Remarks.

Remarks

For columns that return character values, the **Size** property indicates the maximum number of characters that the data source column can hold. For numeric columns, the **Size** property indicates how many bytes of data source storage are required for the column data. This value depends on the data source implementation.

For data source columns that require the use of **GetChunk** and **AppendChunk** methods, the **Size** property is always 0 — you can use the **ColumnSize** method to return correct size information. The maximum size of a *chunk*-type column is limited only by your system resources or the maximum size of the database.

Source Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSourceC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSourceA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSourceS"}

Returns a value that indicates the source of a remote data access error.

Syntax

object.**Source**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Source** property return value is a string expression as described in Remarks.

Remarks

When an error occurs during an ODBC operation, an **rdoError** object is appended to the **rdoErrors** collection. If the error occurred within RDO, the return value begins with "MSRDO20". The object class that caused the error might also be appended to the value of the **Source** property.

SourceColumn Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSourceColumnSourceTableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproSourceColumnSourceTableX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSourceColumnSourceTableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSourceColumnSourceTableS"}

The **SourceColumn** property returns a value that indicates the name of the column that is the original source of the data for an **rdoColumn** object.

This property is not available at design time and is read-only at run time.

Syntax

object.**SourceColumn**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Return Values

The **SourceColumn** property returns a string expression that specifies the name of the column that is the source of data.

Remarks

This property indicates the original column name associated with an **rdoColumn** object. For example, you could use this property to determine the original source of the data in a query column whose name is unrelated to the name of the column in the underlying table.

For columns in **rdoResultset** objects, the **SourceColumn** and **SourceTable** properties return the column name and table name of the base table or the columns and table(s) used to define the query.

SourceTable Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSourceTableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproSourceTableX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSourceTableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSourceTableS"}

SourceTable returns a value that indicates the name of the table that is the original source of the data for an **rdoColumn** object.

This property is not available at design time and is read-only at run time.

Syntax

object.**SourceTable**

The *object* placeholder is an object expression that evaluates to an object in the Applies To list.

Return Values

The **SourceTable** property returns a string expression that specifies the name of the table that is the source of data.

Remarks

This property indicates the original table name associated with an **rdoColumn** object. For example, you could use these properties to determine the original source of the data in a query column whose name is unrelated to the name of the column in the underlying table.

For columns in **rdoResultset** objects, the **SourceColumn** and **SourceTable** properties return the column name and table name of the base table or the columns and table(s) used to define the query.

SQL Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSQLC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproSQLX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSQLA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSQLS"}

Returns or sets the SQL statement that defines the query executed by an **rdoQuery object** or a **RemoteData** control.

Syntax

object.**SQL** [= *value*]

The **SQL** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that contains a value as described in Settings. (Data type is String.)

Settings

The settings for *value* are:

Setting	Description
A valid SQL statement	An SQL query using syntax appropriate for the data source.
A stored procedure	The name of a stored procedure supported by the data source preceded with the keyword "Execute".
An **rdoQuery**	The name of one of the **rdoQuery** objects in the **rdoConnection** object's **rdoQueries** collection.
An **rdoResultset**	The name of one of the **rdoResultset** objects in the **rdoConnection** object's **rdoResultsets** collection.
A table name	The name of one of the populated **rdoTable** objects defined in the **rdoConnection** object's **rdoTables** collection.

Remarks

The **SQL** property contains the structured query language statement that determines how rows are selected, grouped, and ordered when you execute a query. You can use a query to select rows to include in an **rdoResultset** object. You can also define action queries to modify data without returning rows.

You cannot provide a table name at design time for the **SQL** property. However, you can either use a simple query like SELECT * FROM <table>, or at runtime, populate the **rdoTables** collection and use one of the table names returned in the collection. The **rdoTables** collection is populated as soon as it is associated with an active connection and referenced.

The SQL syntax used in a query must conform to the SQL dialect as defined by the data source query processor. The SQL dialect supported by the ODBC interface is defined by the X/Open standard. Generally, a driver scans an SQL statement looking for specific escape sequences that are used to

identify non-standard operands like timestamp literals and functions.

When you need to return rows from a query, you generally provide a SELECT statement in the **SQL** property. The SELECT statement specifies:

- The name of each column to return or "*" to indicate all columns of the specified tables are to be returned. Ambiguous column names must be addressed to include the table name as needed. You can also specify aggregate expressions to perform arithmetic or other functions on the columns selected.
- The name of each table that is to be searched for the information requested. If you specify more than one table, you must provide a WHERE clause to indicate which column(s) are used to cross-reference the information in the tables. Generally, these columns have the same name and meaning. For example the CustomerID column in the Customers table and the Orders table might be referenced.
- (Optionally) a WHERE clause to specify how to join the tables specified and how to limit or filter the number and types of rows returned. You can use parameters in the WHERE clause to specify different sets of information from query to query.
- (Optionally) other clauses such as ORDER BY to set a particular order for the rows or GROUP BY to structure the rows in related sets.

Each SQL dialect supports different syntax and different ancillary clauses. See the documentation provided with your remote server for more details.

Specifying Parameters

If the SQL statement includes question mark parameter markers (?) for the query, you must provide these parameters before you execute the query. Until you reset the parameters, the same parameter values are applied each time you execute the query. To use the **rdoParameters** collection to manage SQL query parameters, you must include the "?" parameter marker in the SQL statement. Input, output, input/output and return value parameters must all be identified in this manner. In some cases, you must use the **Direction** property to indicate how the parameter will be used.

Note When executing stored procedures that do *not* require parameters, do not include the parenthesis in the SQL statement. For example, to execute the "MySP" procedure use the following syntax: `{Call MySP }`.

Note When using Microsoft SQL Server 6 as a data source, the ODBC driver automatically sets the **Direction** property. You also do not need to set the **Direction** property for input parameters, as this is the default setting.

If the user changes the parameter value, you can re-apply the parameter value and re-execute the query by using the **Requery** method against the **rdoResultset** (`MyRs`).

```
Cpw(0) = Text1.Text
MyRs.Requery
```

You can also specify parameters in any SQL statement by concatenating the parameters to the SQL statement string. For example, to submit a query using this technique, you can use the following code:

```
QSQL$ = "SELECT * FROM Authors WHERE Au_Lname = '" _
 & Text.Text & "'"
Set CPw = cn.CreateQuery("",QSQL$)
Set MyRs = Cpw.OpenResultSet()
```

In this case, the **rdoParameters** collection is *not* created and cannot be referenced. To change the query parameter, you must rebuild the SQL statement with the new parameter value each time the query is executed, or before you use the **Requery** method.

The SQL statement may include an ORDER BY clause to change the order of the rows returned by the **rdoResultset** or a WHERE clause to filter the rows.

Note You can't use the **rdoTable** object names until the **rdoTables** collection is referenced. When your code references the **rdoTables** collection by enumerating one or more of its members, RDO queries the data source for table meta data. This results in population of the **rdoTables** collection. This means that you cannot simply provide a table name for the *value* argument without first enumerating the **rdoTables** collection.

RemoteData Control

When used with the **RemoteData** control, the **SQL** property specifies the source of the data rows accessible through bound controls on your form.

If you set the **SQL** property to an SQL statement that returns rows or to the name of an existing **rdoQuery**, all columns returned by the **rdoResultset** are visible to the bound controls associated with the **RemoteData** control.

After changing the value of the **SQL** property at run time, you must use the **Refresh** method to activate the change.

Note Whenever your **rdoQuery** or SQL statement returns a value from an expression, the column name of the expression is determined by the wording of the SQL query. In most cases you'll want to alias expressions so you know the name of the column to bind to the bound control.

Make sure each bound control has a valid setting for its **DataField** property. If you change the setting of a **RemoteData** control's **SQL** property and then use **Refresh**, the **rdoResultset** identifies the new object. This may invalidate the **DataField** settings of bound controls and cause a trappable error.

SQL Property Example

For example, to execute a procedure that accepts two input parameters and returns a return value and an output parameter, you can use the following code. The example creates an **rdoQuery** object whose SQL property is set to the string specified by QSQL$. This query calls a stored procedure that returns a return status, and an output argument as well as accepting two input arguments.

```
Dim Cqy as new rdoQuery
Dim MyRs as rdoResultset
Cqy.SQL = "{ ? = call sp_MyProc (?, ?) }"
Cqy(0).Direction = rdReturnValue
Cqy(1).Direction = rdParamInput
Cqy(2).Direction = rdParamInput
Cqy(3).Direction = rdParamOutput
Cqy(1) = "Victoria"
Cqy(0) = 21
Set MyRs = Cqy.OpenResultSet(rdOpenForwardOnly)
```

SQLRetCode Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSqlretcodeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSqlretcodeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSqlretcodeS"}

Returns the ODBC error return code from the most recent RDO operation.

Syntax

object.**SQLRetCode**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **SQLRetCode** property return value is a Long value that corresponds to one of the following constants:

Constant	Value	Description
rdSQLSuccess	0	The operation is successful.
rdSQLSuccessWithInfo	1	The operation is successful, and additional information is available.
rdSQLNoDataFound	100	No additional data is available.
rdSQLError	-1	An error occurred performing the operation.
rdSQLInvalidHandle	-2	The handle supplied is invalid.

SQLState Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproSqlstateC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproSqlstateA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproSqlstateS"}

Returns a value corresponding to the type of error as defined by the X/Open and SQL Access Group SQL.

Syntax

object.**SQLState**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **SQLState** return value is a five-character string expression, as described in Remarks.

Remarks

When an RDO operation returns an error, or completes an operation, the **SQLState** property of the **rdoError** object is set. If the error is not caused by ODBC or if no **SQLState** is available, the **SQLState** property returns an empty string.

The character string value returned by the **SQLState** property consists of a two-character class value followed by a three-character subclass value. A class value of "01" indicates a warning and is accompanied by a return code of **rdSQLSuccessWithInfo**.

Class values other than "01", except for the class "IM", indicate an error and are accompanied by a return code of **rdSQLError**. The class "IM" is specific to warnings and errors that derive from the implementation of ODBC itself. The subclass "000" in any class is for implementation-defined conditions within the given class. The assignment of class and subclass values is defined by ANSI SQL-92.

StillExecuting Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproStillExecutingC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproStillExecutingX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproStillExecutingA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproStillExecutingS"}

Returns a Boolean value that indicates whether a query is still executing.

Syntax

object.**StillExecuting**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **StillExecuting** property return values are:

Value	Description
True	The query is still executing.
False	The query is ready to return the result set.

Remarks

Use the **StillExecuting** property to determine if a query is ready to return the first result set. Until the **StillExecuting** property is **False**, the associated object cannot be accessed. However, unless you use the **rdAsyncEnable** option, your application will block until the query is completed and ready to process the result set.

Once the **StillExecuting** property returns **False**, the first or next result set is ready for processing. When you use the **MoreResults** method to complete processing of a result set, the **StillExecuting** property is reset to **True** while the next result sets is retrieved.

The **StillExecuting** property also changes to **True** when you execute a Move method. For example executing **MoveLast** against an **rdoResultset** resets the **StillExecuting** property to **True** as long as RDO continues to fetch rows from the remote server.

You can also use the **QueryComplete** event to indicate when a query has completed and the associated **rdoResultset** object is ready to process.

Use the **Cancel** method to terminate processing of an executing query, including all statements in a batch query.

StillExecuting Property Example

This example illustrates use of the **StillExecuting** property to monitor the progress of a query that is expected to take more than a few seconds. By enabling RDO's asynchronous mode, control returns to the application long before the query is complete. While waiting for the **StillExecuting** property to return **False**, we display a progress bar that has been programmed to reflect the length of time that the query is expected to take. Note that if this time is exceeded, the progress bar is re-calibrated to reflect the longer duration.

```
Dim rdoCn As New rdoConnection
Dim rdoRs As rdoResultset
Dim SQL As String
Dim TimeExpected As Single
Dim Ts As Single, Tn As Single

With rdoCn
    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=SEQUEL;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .EstablishConnection rdDriverNoPrompt, True

    SQL = "Execute VeryLongProcedure"
    TimeExpected = 20 ' We expect this to take 60 sec.

    Set rdoRs = .OpenResultset(Name:=SQL, _
        Type:=rdOpenForwardOnly, _
        LockType:=rdConcurReadOnly, _
        Option:=rdAsyncEnable)
    Ts = Timer
    ProgressBar1.Max = TimeExpected
    While rdoRs.StillExecuting
        Tn = Int(Timer - Ts)
        If Tn < TimeExpected Then
            ProgressBar1 = Tn
        Else
            ProgressBar1.Max = ProgressBar1.Max + 10
            TimeExpected = ProgressBar1.Max
        End If
        DoEvents
    Wend
    Status = "Query done. Duration:" & Int(Timer - Ts)

End With
rdoRs.Close
rdoCn.Close
```

Transactions Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproTransactionsC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproTransactionsA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproTransactionsS"}

Returns a value that indicates whether an object supports the recording of a series of changes that can later be rolled back (undone) or committed (saved).

Syntax

object.**Transactions**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Transactions** property return values are:

Value	Description
True	The object supports transactions.
False	The object doesn't support transactions.

Remarks

Check the **Transactions** property before using the **BeginTrans** method to make sure that transactions are supported. When **Transactions** is **False,** using the **BeginTrans**, **CommitTrans**, or **RollbackTrans** method has no effect.

The **Transactions** property calls the ODBC **SQLGetInfo** function to determine if the ODBC driver is *capable* of supporting transactions, not if the current result set is updatable. You can always call the **BeginTrans** method on the **rdoConnection** object if the **Transactions** property is **True** — even for read-only **rdoResultset** objects.

Type Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproTypeC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproTypeA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproTypeS"}

Returns or sets a value that indicates the type or data type of an object.

Syntax

object.**Type** [= *value*]

The **Type** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A constant or Integer value that specifies a datatype, as described in Return Values.

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

For an **rdoColumn** or **rdoParameter** object, the **Type** property returns an Integer. You can also set the **Type** property on the **rdoParameter** object to indicate the datatype of a specific procedure argument. The valid values are:

Constant	Value	Description
rdTypeCHAR	1	Fixed-length character string. Length set by **Size** property.
rdTypeNUMERIC	2	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeDECIMAL	3	Signed, exact, numeric value with precision p and scale s (1 p 15; 0 s p).
rdTypeINTEGER	4	Signed, exact numeric value with precision 10, scale 0 (signed: -231 n 231-1; unsigned: 0 n 232-1).
rdTypeSMALLINT	5	Signed, exact numeric value with precision 5, scale 0 (signed: -32,768 n 32,767, unsigned: 0 n 65,535).
rdTypeFLOAT	6	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10-308 to 10308).
rdTypeREAL	7	Signed, approximate numeric value with mantissa precision 7 (zero or absolute value 10-38 to 1038).
rdTypeDOUBLE	8	Signed, approximate numeric value with mantissa precision 15 (zero or absolute value 10-308 to 10308).
rdTypeDATE	9	Date — data source dependent.
rdTypeTIME	10	Time — data source dependent.

rdTypeTIMESTAMP	11	TimeStamp — data source dependent.
rdTypeVARCHAR	12	Variable-length character string. Maximum length 255.
rdTypeLONGVARCHAR	-1	Variable-length character string. Maximum length determined by data source.
rdTypeBINARY	-2	Fixed-length binary data. Maximum length 255.
rdTypeVARBINARY	-3	Variable-length binary data. Maximum length 255.
rdTypeLONGVARBINARY	-4	Variable-length binary data. Maximum data source dependent.
rdTypeBIGINT	-5	Signed, exact numeric value with precision 19 (signed) or 20 (unsigned), scale 0; (signed: -263 n 263-1; unsigned: 0 n 264-1).
rdTypeTINYINT	-6	Signed, exact numeric value with precision 3, scale 0; (signed: -128 n 127, unsigned: 0 n 255).
rdTypeBIT	-7	Single binary digit.

For an **rdoQuery** object, the **Type** property returns an Integer. The return values are:

Constant	Value	Query type
rdQSelect	0	Select
rdQAction	1	Action
rdQProcedures	2	Procedural
rdQCompound	3	The query contains both action and select statements

For an **rdoResultset** object, the **Type** property returns an Integer that determines the type of **rdoResultset**. The return values are:

Constant	Value	rdoResultset type
rdOpenForwardOnly	0	Fixed set, non-scrolling.
rdOpenKeyset	1	Updatable, fixed set, scrollable query result set cursor.
rdOpenDynamic	2	Updatable, dynamic set, scrollable query result set cursor.
rdOpenStatic	3	Read-only, fixed set.

Note Not all ODBC drivers or data sources support every type of **rdoResultset** cursor type. If you choose a cursor that is not supported, the ODBC driver attempts to revert to a supported type. If no supported type is available, a trappable error is fired.

For an **rdoTable** object, the **Type** property returns a String. The settings for *value* are determined by the data source driver.

Typically, this string value is "TABLE", "VIEW", "SYSTEM TABLE", "GLOBAL TEMPORARY". "LOCAL TEMPORARY", "ALIAS", "SYNONYM" or some other data source-specific type identifier.

Remarks

Depending on the object, the **Type** property indicates:

Object	Type indicates
rdoColumn, rdoParameter	Object data type
rdoQuery	Type of query
rdoResultset	Type of **rdoResultset**
rdoTable	Type of table on data source

In some cases, you must override the **Type** property assignment made by RDO when creating some types of parameter queries. For example, if a parameter is passed to an expression inside of an SQL statement, the ODBC driver might not be able to determine the correct type. In these cases, you can force a specific parameter to be handled as the correct type by simply setting the **rdoParameter** object's **Type** property. This is the only situation that permits you to change the **Type** property. In all other cases, this property is read-only.

Updatable Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproUpdatableC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproUpdatableA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproUpdatableS"}

Returns a Boolean value that indicates whether changes can be made to a remote data object.

Syntax

object.**Updatable**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Updatable** property return values are:

Value	Description
True	The object can be changed or updated.
False	The object can't be changed or updated. This is the only setting for static-type **rdoResultset** objects.

Remarks

If the **Updatable** property setting is **True**, the specified:

- **rdoConnection** object refers to an updatable data source.
- **rdoQuery** object refers to an updatable result set.
- **rdoResultset** contains updatable rows.
- **rdoTable** object refers to a table whose data can be changed through use of a query.
- **rdoColumn** object refers to data that can be changed. Only **rdoColumn** objects which are part of an **rdoResultset** object can be changed.

You can use the **Updatable** property with all types of **rdoResultset** objects.

Many types of **rdoResultset** objects can contain columns that can't be updated. For example, you can create a forward-only **rdoResultset** that is derived from nonupdatable sources or that contains computed or derived columns.

If the object contains only nonupdatable columns, the value of the **Updatable** property is **False**. When one or more columns are updatable, the property's value is **True**. You can edit only the updatable columns. A trappable error occurs if you try to assign a new value to a nonupdatable column.

Because an updatable object can contain columns that cannot be updated, check the **Updatable** property of each **rdoColumn** before editing a row in the **rdoResultset**.

Even when a cursor cannot be updated, it might still be possible to update the data through use of an action query. In many cases, database tables are protected and not updatable by design — as they are protected from direct access by the system administrator and the remote system's permission scheme. If this is the case, check with your system administrator for the availability of stored procedures or special login accounts that permit you to perform your changes.

UserName Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproUserNameC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproUserNameA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproUserNameS"}

Returns or sets a value that represents a user of an **rdoEnvironment** object. Use the **UserName** property with the **Password** property to connect to an ODBC data source.

Syntax

object.**UserName** [= *value*]

The **UserName** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A string expression that contains a user name as described in Settings. (Data type is String.)

Settings

The user name syntax depends on the ODBC data source.

Return Values

The **UserName** property represents the user of an **rdoEnvironment** object. The user name is set when the **rdoEnvironment** is either created automatically by the **RemoteData** control, by the first reference to a remote data object, or when the **rdoCreateEnvironment** method is executed.

You can determine the default user name with the **rdoDefaultUser** property of the **rdoEngine** object. If no specific user name is supplied in **UserName**, the value of the **rdoDefaultUser** property is used.

Value Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproValueC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproValueA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproValueS"}

Returns or sets the value of an object.

Syntax

object.**Value** [= *value*]

The **Value** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An expression that evaluates to a value appropriate for the data type, as specified by the **Type** property of an object. (Data type is Variant.)

Remarks

Use the **Value** property to retrieve and alter data in **rdoResultset** objects. The data type of the data returned is indicated by the **Type** property of the object.

The **Value** property is the default property of the **rdoColumn** and **rdoParameter** objects. Therefore, the following lines of code are equivalent (assuming Column1 is at the first ordinal position):

```
Dim MyResultset As rdoResultset
X = MyResultset!Column1
X = MyResultset!Column1.Value
X = MyResultset(0)
X = MyResultset(0).Value
X = MyResultset("Column1").Value
X = MyResultset("Column1")
X = RemoteData1.Resultset("Column1")
X = RemoteData1.Resultset(0)
F$ = "Column1" : X = MyResultset(F$).Value
X = MyResultset(F$)
Set X = MyResultset(0): X.Value : X
```

Version Property (Remote Data)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproVersionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproVersionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproVersionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproVersionS"}

Returns a value that indicates the version of the data source associated with the object.

Syntax

object.**Version**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **Version** property return value is a 10-character string expression.

Remarks

For an **rdoConnection** object, this property identifies the version of the data source used when the connection was created. This value is the version of ODBC to which the driver manager conforms. The version is in the form ##.##.####, where the first two digits are the major version number, the next two digits are the minor version, and the last four digits are the build number.

ActiveConnection Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproActiveConnectionC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproActiveConnectionX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproActiveConnectionA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproActiveConnectionS"}

Returns or sets an object reference indicating the connection this query should be associated with.

Syntax

object.**ActiveConnection** [= *value*]

The **ActiveConnection** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An expression that evaluates to a valid **rdoConnection** or derived object. *Value* defaults to the **rdoConnection** used to create the object or **Nothing**.

Remarks

The **ActiveConnection** property holds a reference to the connection associated with the **rdoQuery** or **rdoResultset** object. All database statements executed by the object are executed against this connection.

When working with an **rdoQuery** object, the **ActiveConnection** property can be set to **Nothing** which dissociates the object from a specific connection. You can subsequently re-associate the **rdoQuery** object to another **rdoConnection** object by setting the **ActiveConnection** object. Using this technique, a query can be executed against a set of connections.

When working with the **rdoResultset** object and the Client Batch cursor library, you can set the **ActiveConnection** property to **Nothing**. In this case, if the result set is created with a static cursor and the **rdConcurBatch** concurrency option, the **rdoResultset** data is still available and you are free to make changes or additions to the result set. Once you set the **ActiveConnection** to an open **rdoConnection** object, you can use the **BatchUpdate** method to post these changes to the remote database.

ActiveConnection Property Example

The following examples illustrates use of the **ActiveConnection** property to select an **rdoConnection**. In this case, the application opens two separate connections and uses the same **rdoQuery** against each.

```
Dim rdoCn As New rdoConnection
Dim rdoCn2 As New rdoConnection
Dim rdoQy As New rdoQuery
Dim rdoRs As rdoResultset
Dim rdoCol As rdoColumn
Dim rdoEn As rdoEnvironment

Private Sub Form_Load()
On Error GoTo CnEh

Set rdoEn = rdoEnvironments(0)

With rdoCn

    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=Betav486;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .EstablishConnection rdDriverNoPrompt, True
    rdoEn.rdoConnections.Add rdoCn
End With

With rdoCn2
    .Connect = "UID=;PWD=;Database=Pubs;" _
        & "Server=Betav486;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 5
    .EstablishConnection rdDriverNoPrompt, True
    rdoEn.rdoConnections.Add rdoCn2
End With

With rdoQy
    Set .ActiveConnection = rdoCn
    .SQL = "Select Name, refDate " _
      & " from Sysobjects where type = 'U' "
    .LockType = rdConcurReadOnly
    .RowsetSize = 1
    .CursorType = rdUseServer
End With

For Each rdoCn In rdoEn.rdoConnections
    Set rdoQy.ActiveConnection = rdoCn
    Set rdoRs = rdoQy.OpenResultset(rdOpenForwardOnly)
    With rdoRs
        For Each rdoCol In rdoRs.rdoColumns
            Debug.Print rdoCol.Name,
        Next
        Debug.Print
        Do Until rdoRs.EOF
                For Each rdoCol In rdoRs.rdoColumns
                    Debug.Print rdoCol
```

```
                Next
            rdoRs.MoveNext
        Loop
    End With
Next                    ' Next Connection

Exit Sub
CnEh:
Dim er As rdoError
    Debug.Print Err, Error
    For Each er In rdoErrors
        Debug.Print er.Description, er.Number
    Next er
    Resume Next
End Sub
```

StillConnecting Property

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproStillConnectingC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"rdproStillConnectingX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproStillConnectingA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproStillConnectingS"}

Returns a value that indicates if the connection has been established.

Syntax

object.**StillConnecting**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Return Values

The **StillConnecting** property return values are:

Value	Description
True	The connection is being made asynchronously but has not been established.
False	The connection has been established.

Remarks

This property works very similarly to the **StillExecuting** property, except that it is **True** while an asynchronous connection to the server is being performed. This property is set to **False** again after the connection has been established.

All method and property access of the connection object (with the exception of this property and the **Cancel** method) will result in trappable errors while an asynchronous connection is in progress.

StillConnecting Property Example

This example illustrates use of the **StillConnecting** property when establishing a connection using the **rdAsyncEnable** option.

```
Dim rdoCn As New rdoConnection
Dim TimeExpected As Single
Dim Ts As Single, Tn As Single

TimeExpected = 15
With rdoCn
    .Connect = "UID=;PWD=;Database=WorkDB;" _
        & "Server=FarAway;Driver={SQL Server}" _
        & "DSN='';"
    .LoginTimeout = 45
    .EstablishConnection rdDriverNoPrompt, _
       True, rdAsyncEnable
    Ts = Timer
    ProgressBar1.Max = TimeExpected  ' time to Open
    While .StillConnecting
        Tn = Int(Timer - Ts)
        If Tn < TimeExpected Then
            ProgressBar1 = Tn
        Else
            ProgressBar1.Max = ProgressBar1.Max + 10
            TimeExpected = ProgressBar1.Max
        End If
        DoEvents
    Wend
    Status = "Duration:" & Int(Timer - Ts)

End With
rdoCn.Close
Exit Sub
```

Item Property (RDO)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"rdproItemC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"rdproItemA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"rdproItemS"}

Returns a specific member of an Remote Data Objects (RDO) collection object either by position or by key.

Syntax

object.**Item**(*index*)

The **Item** property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Required. An expression that specifies the position of a member of the collection. If a numeric expression, index must be a number from 1 to the value of the collection's **Count** property. If a String expression, index must correspond to the key argument specified when the member referred to was added to the collection.

Remarks

Basically, the **Item** property is used to choose a member of the collection either by ordinal number or by a key value.

If the value provided as *index* doesn't match any existing member of the collection, an error occurs.

The **Item** property is the default method for a collection and is rarely used to reference collection members. Therefore, the following lines of code are equivalent:

```
Print MyCollection(1)
Print MyCollection.Item(1)
```

Change Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtChangeEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtChangeEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtChangeEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtChangeEventControlsPlaceholderS"}

Click Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtClickEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtClickEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtClickEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtClickEventControlsPlaceholderS"}

DblClick Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDblClickEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDblClickEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDblClickEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDblClickEventControlsPlaceholderS"}

DragDrop Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragDropEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragDropEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragDropEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragDropEventControlsPlaceholderS"}

DragOver Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtDragOverEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtDragOverEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtDragOverEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtDragOverEventControlsPlaceholderS"}

KeyDown Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyDownEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyDownEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyDownEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyDownEventControlsPlaceholderS"}

KeyPress Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyPressEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyPressEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyPressEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyPressEventControlsPlaceholderS"}

KeyUp Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtKeyUpEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtKeyUpEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtKeyUpEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtKeyUpEventControlsPlaceholderS"}

MouseDown Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseDownEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseDownEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseDownEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseDownEventControlsPlaceholderS"}

MouseMove Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseMoveEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseMoveEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseMoveEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseMoveEventControlsPlaceholderS"}

MouseUp Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtMouseUpEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtMouseUpEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtMouseUpEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtMouseUpEventControlsPlaceholderS"}

OLECompleteDragEvent (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLECompleteDragEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLECompleteDragEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLECompleteDragEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLECompleteDragEventControlsPlaceholderS"}

OLEDragDrop Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragDropEventPHolderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragDropEventPHolderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragDropEventPHolderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragDropEventPHolderS"}

OLEDragOver Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEDragOverEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEDragOverEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEDragOverEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEDragOverEventControlsPlaceholderS"}

OLEGiveFeedback Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEGiveFeedbackEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEGiveFeedbackEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEGiveFeedbackEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEGiveFeedbackEventControlsPlaceholderS"}

OLESetData Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLESetDataEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLESetDataEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLESetDataEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLESetDataEventControlsPlaceholderS"}

OLEStartDrag Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtOLEStartDragEventControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtOLEStartDragEventControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtOLEStartDragEventControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtOLEStartDragEventControlsPlaceholderS"}

Refresh Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRefreshMethodControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRefreshMethodControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRefreshMethodControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRefreshMethodControlsPlaceholderS"}

OLEDrag Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthOLEDragMethodControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthOLEDragMethodControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthOLEDragMethodControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthOLEDragMethodControlsPlaceholderS"}

Appearance Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAppearancePropertyControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAppearancePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAppearancePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAppearancePropertyControlsPlaceholderS"}

BackColor Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackColorPropertyControlsPlaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackColorPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackColorPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackColorPropertyControlsPlaceholderS"}

Caption Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCaptionPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCaptionPropertyControlsPlaceholderS"}

Enabled Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproEnabledPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnabledPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnabledPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnabledPropertyControlsPlaceholderS"}

Font Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontPropertyControlsPlaceholderS"}

ForeColor Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproForeColorPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproForeColorPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproForeColorPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproForeColorPropertyControlsPlaceholderS"}

IntegralHeight Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIntegralHeightPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIntegralHeightPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIntegralHeightPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIntegralHeightPropertyControlsPlaceholderS"}

Locked Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLockedPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLockedPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLockedPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLockedPropertyControlsPlaceholderS"}

MouseIcon Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMouseIconPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseIconPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseIconPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseIconPropertyControlsPlaceholderS"}

MousePointer Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMousePointerPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerPropertyControlsPlaceholderS"}

OLEDragMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEDragModePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragModePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragModePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragModePropertyControlsPlaceholderS"}

OLEDropMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproOLEDropModePropertyControlsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropModePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropModePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropModePropertyControlsPlaceholderS"}

SelLength Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelLengthPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelLengthPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelLengthPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelLengthPropertyControlsPlaceholderS"}

SelStart Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelStartPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelStartPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelStartPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelStartPropertyControlsPlaceholderS"}

SelText Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproSelTextPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelTextPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelTextPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelTextPropertyControlsPlaceholderS"}

Style Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproStylePropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproStylePropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproStylePropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproStylePropertyControlsPlaceholderS"}

Text Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTextPropertyControlsPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyControlsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyControlsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyControlsPlaceholderS"}

Keyword Not Found

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic"}

The keyword you've selected can't be found in Visual Basic Help. You may have misspelled the keyword, selected too much or too little text, or asked for help on a word that is not a valid Visual Basic keyword.

The easiest way to get help on a specific keyword is to position the insertion point anywhere within the keyword you want help on and press F1. You do not need to select the keyword. In fact, if you select only a portion of the keyword, or more than a single word, Help will not find what you're looking for.

To use the built-in Help search dialog box, press the **Help Topics** button on the toolbar.

Left, Top Properties (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproLeftTopPropertiesPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftTopPropertiesPlaceholderS"}

Height, Width Properties (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHeightWidthPropertiesPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeightWidthPropertiesPlaceholderS"}

Tag Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTagPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTagPropertyPlaceholderS"}

ToolTipText Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproToolTipTextPropertyPlaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproToolTipTextPropertyPlaceholderS"}

Count Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCountPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCountPropertyPlaceholderS"}

Item Method (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproItemMethodPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthItemMethodPlaceholderS;vbproItemMethodPlaceholderS"}

Hwnd Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHwndPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHwndPropertyPlaceholderS"}

TabStop Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabStopPropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTabStopPropertyPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTabStopPropertyPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabStopPropertyPlaceholderS"}

Visible Property (Placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproVisiblePropertyPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVisiblePropertyPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVisiblePropertyPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVisiblePropertyPlaceholderS"}

Picture Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPicturePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPicturePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPicturePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPicturePropertyplaceholderS"}

Max, Min Properties (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMaxMinPropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxMinPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxMinPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxMinPropertiesplaceholderS"}

Add Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthAddMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthAddMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthAddMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthAddMethodplaceholderS"}

Clear Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearMethodplaceholderS"}

Container Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproContainerPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproContainerPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproContainerPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproContainerPropertyplaceholderS"}

Controls Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolControlsCollectionplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolControlsCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbcolControlsCollectionplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbcolControlsCollectionplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbcolControlsCollectionplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolControlsCollectionplaceholderS"}

Copies Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproCopiesPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCopiesPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCopiesPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCopiesPropertyplaceholderS"}

DataBinding Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbobjDataBindingObjectplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbobjDataBindingObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Properties":"vbobjDataBindingObjectplaceholderP"} {ewc HLP95EN.DLL,DYNALINK,"Methods":"vbobjDataBindingObjectplaceholderM"} {ewc HLP95EN.DLL,DYNALINK,"Events":"vbobjDataBindingObjectplaceholderE"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataBindingObjectplaceholderS"}

DataBindings Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataBindingsCollectionplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataBindingsCollectionplaceholderX":1} {ewc
HLP95EN.DLL,DYNALINK,"Properties":"vbcolDataBindingsCollectionplaceholderP"} {ewc
HLP95EN.DLL,DYNALINK,"Methods":"vbcolDataBindingsCollectionplaceholderM"} {ewc
HLP95EN.DLL,DYNALINK,"Events":"vbcolDataBindingsCollectionplaceholderE"} {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataBindingsCollectionplaceholderS"}

Filename Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFilenamePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFilenamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFilenamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFilenamePropertyplaceholderS"}

FontBold, FontItalic, FontStrikeThru Properties (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontBoldFontItalicFontStrikeThruPropertiesplaceholderS"}

FontName Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontNamePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontNamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontNamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontNamePropertyplaceholderS"}

FontSize Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFontSizePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontSizePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontSizePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontSizePropertyplaceholderS"}

GetData Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetDataMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetDataMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetDataMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetDataMethodplaceholderS"}

GetFormat Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthGetFormatMethodplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthGetFormatMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthGetFormatMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthGetFormatMethodplaceholderS"}

HelpFile Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHelpFilePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHelpFilePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHelpFilePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpFilePropertyplaceholderS"}

Item Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproItemPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproItemPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproItemPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItemPropertyplaceholderS"}

Remove Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodplaceholderS"}

SetData Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetDataMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetDataMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetDataMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetDataMethodplaceholderS"}

Align Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAlignPropertyplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAlignPropertyplaceholderS"}

DragIcon Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDragIconPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragIconPropertyplaceholderS"}

DragMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDragModePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragModePropertyplaceholderS"}

HelpContextID Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHelpContextIDPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpContextIDPropertyplaceholderS"}

TabIndex Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTabIndexPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTabIndexPropertyplaceholderS"}

Alignment Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstAlignmentConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Border Property Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstBorderPropertyConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

BorderStyle Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstBorderStyleConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

BorderStyle Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproBorderStylePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStylePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStylePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStylePropertyActiveXControlsplaceholderS"}

Clear Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthClearMethodActiveXControlsplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthClearMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthClearMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthClearMethodActiveXControlsplaceholderS"}

Clipboard Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstClipboardObjectConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Color Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstColorConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

CommonDialog Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstCommonDialogControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

CommonDialog Error Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstCommonDialogErrorConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

DataBindings Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataBindingsPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataBindingsPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataBindingsPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataBindingsPropertyplaceholderS"}

DDE Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDDEConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Drag-and-Drop Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDragandDropConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Drawing Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstDrawingConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

FetchVerbs Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFetchVerbsMethodC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFetchVerbsMethodX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFetchVerbsMethodA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFetchVerbsMethodS"}

Form Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstFormConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Graphics Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstGraphicsConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Grid Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstGridControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Help Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstHelpConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

HideSelection Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideSelectionPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionPropertyActiveXControlsplaceholderS"}

HideSelection Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHideSelectionPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionPropertyplaceholderS"}

Image Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImagePropertyActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImagePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImagePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImagePropertyActiveXControlsplaceholderS"}

ImageList Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproImageListPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproImageListPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproImageListPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproImageListPropertyActiveXControlsplaceholderS"}

Index Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexPropertyActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyActiveXControlsplaceholderS"}

Index Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyplaceholderS"}

Key Code Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstKeyCodeConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Key Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproKeyPropertyActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyPropertyActiveXControlsplaceholderS"}

Menu Accelerator Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMenuAcceleratorConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Menu Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMenuControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Miscellaneous Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMiscellaneousConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

MousePointer Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstMousePointerConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

OLE Container Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstOLEContainerControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Picture Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstPictureObjectConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Printer Object Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstPrinterObjectConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

RasterOp Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstRasterOpConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Remove Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodActiveXControlsplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodActiveXControlsplaceholderS"}

ShowInTaskbar Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstShowInTaskbarPropertyplaceholderC;vbproBooksOnlineJumpTopic"}

ShowTips Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproShowTipsPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowTipsPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowTipsPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowTipsPropertyActiveXControlsplaceholderS"}

ShowWhatsThis Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthShowWhatsThisMethodplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthShowWhatsThisMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthShowWhatsThisMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthShowWhatsThisMethodplaceholderS"}

Text Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproTextPropertyActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyActiveXControlsplaceholderS"}

Value Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproValuePropertyActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValuePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValuePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValuePropertyActiveXControlsplaceholderS"}

Variant Type Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstVariantTypeConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

Visual Basic Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstVisualBasicConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

WhatsThisButton Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisButtonPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisButtonPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisButtonPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisButtonPropertyplaceholderS"}

WhatsThisHelp Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisHelpPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpPropertyplaceholderS"}

WhatsThisHelpID Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproWhatsThisHelpIDPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproWhatsThisHelpIDPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproWhatsThisHelpIDPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproWhatsThisHelpIDPropertyplaceholderS"}

WhatsThisMode Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthWhatsThisModeMethodplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthWhatsThisModeMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthWhatsThisModeMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthWhatsThisModeMethodplaceholderS"}

Windows 95 Control Constants (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcstWindows95ControlConstantsplaceholderC;vbproBooksOnlineJumpTopic"}

ColIsVisible Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColIsVisiblePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColIsVisiblePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColIsVisiblePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColIsVisiblePropertyplaceholderS"}

ColPos Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproColPosPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproColPosPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproColPosPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproColPosPropertyplaceholderS"}

DataObject Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataObjectObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataObjectObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataObjectObjectplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbobjDataObjectObjectplaceholderS;vbproDataObjectObjectplaceholderS"}

DataObjectFiles Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolDataObjectFilesCollectionplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolDataObjectFilesCollectionplaceholderX":1} {ewc
HLP95EN.DLL,DYNALINK,"Properties":"vbcolDataObjectFilesCollectionplaceholderP"} {ewc
HLP95EN.DLL,DYNALINK,"Methods":"vbcolDataObjectFilesCollectionplaceholderM"} {ewc
HLP95EN.DLL,DYNALINK,"Events":"vbcolDataObjectFilesCollectionplaceholderE"} {ewc
HLP95EN.DLL,DYNALINK,"Specifics":"vbcolDataObjectFilesCollectionplaceholderS"}

DataSource Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDataSourcePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataSourcePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataSourcePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataSourcePropertyplaceholderS"}

Drag Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthDragMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthDragMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthDragMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthDragMethodplaceholderS"}

DrawMode Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproDrawModePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDrawModePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDrawModePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDrawModePropertyplaceholderS"}

FixedAlignment Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproFixedAlignmentPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFixedAlignmentPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFixedAlignmentPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFixedAlignmentPropertyplaceholderS"}

GotFocus Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtGotFocusEventplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtGotFocusEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtGotFocusEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtGotFocusEventplaceholderS"}

GridLineWidth Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproGridLineWidthPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGridLineWidthPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGridLineWidthPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGridLineWidthPropertyplaceholderS"}

Index Property (Control Array) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproIndexPropertyControlArrayplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproIndexPropertyControlArrayplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproIndexPropertyControlArrayplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproIndexPropertyControlArrayplaceholderS"}

LostFocus Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtLostFocusEventplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtLostFocusEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtLostFocusEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtLostFocusEventplaceholderS"}

Move Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthMoveMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthMoveMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthMoveMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthMoveMethodplaceholderS"}

Name Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproNamePropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNamePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNamePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNamePropertyplaceholderS"}

Object Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproObjectPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproObjectPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproObjectPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproObjectPropertyplaceholderS"}

Parent Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproParentPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproParentPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproParentPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproParentPropertyplaceholderS"}

RowColChange Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbevtRowColChangeEventplaceholderC;vbproBooksOnlineJumpTopic"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbevtRowColChangeEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbevtRowColChangeEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbevtRowColChangeEventplaceholderS"}

RowIsVisible Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRowIsVisiblePropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRowIsVisiblePropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRowIsVisiblePropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRowIsVisiblePropertyplaceholderS"}

RowPos Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRowPosPropertyplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRowPosPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRowPosPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRowPosPropertyplaceholderS"}

SetFocus Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthSetFocusMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthSetFocusMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthSetFocusMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthSetFocusMethodplaceholderS"}

ZOrder Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthZOrderMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthZOrderMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthZOrderMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthZOrderMethodplaceholderS"}

Files Method (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthFilesMethodplaceholderC;vbproBooksOnlineJumpTopic"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthFilesMethodplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthFilesMethodplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthFilesMethodplaceholderS"}

Connect Event (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproConnectEventplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproConnectEventplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproConnectEventplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproConnectEventplaceholderS"}

HScrollSmallChange, VScrollSmallChange Properties (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproHScrollSmallChangeVScrollSmallChangePropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHScrollSmallChangeVScrollSmallChangePropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHScrollSmallChangeVScrollSmallChangePropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHScrollSmallChangeVScrollSmallChangePropertiesplaceholderS"}

MinHeight, MinWidth Properties (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproMinHeightMinWidthPropertiesplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMinHeightMinWidthPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMinHeightMinWidthPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMinHeightMinWidthPropertiesplaceholderS"}

Property Pages Dialog Box (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproPropertryPagesDialogBoxplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertryPagesDialogBoxplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertryPagesDialogBoxplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertryPagesDialogBoxplaceholderS"}

RemoteHost Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRemoteHostPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemoteHostPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemoteHostPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemoteHostPropertyplaceholderS"}

RemotePort Property (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproRemotePortPropertyplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemotePortPropertyplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemotePortPropertyplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemotePortPropertyplaceholderS"}

ViewportHeight, ViewportLeft, ViewportTop, ViewportWidth Properties (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBooksOnlineJumpTopic;vbproViewportHeightViewportLeftViewportTopViewportWidthPropertiesplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproViewportHeightViewportLeftViewportTopViewportWidthPropertiesplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproViewportHeightViewportLeftViewportTopViewportWidthPropertiesplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproViewportHeightViewportLeftViewportTopViewportWidthPropertiesplaceholderS"}

Resync Method (Remote Data) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthResyncMethodRemoteDataplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthResyncMethodRemoteDataplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthResyncMethodRemoteDataplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthResyncMethodRemoteDataplaceholderS"}

ContainedVBControls Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbcolContainedVBControlsCollectionplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbcolContainedVBControlsCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbcolContainedVBControlsCollectionplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbcolContainedVBControlsCollectionplaceholderS"}

Member Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMemberObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMemberObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMemberObjectplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMemberObjectplaceholderS"}

Members Collection (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMembersCollectionplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMembersCollectionplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMembersCollectionplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMembersCollectionplaceholderS"}

VBComponentsEvents Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproVBComponentsEventsObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVBComponentsEventsObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVBComponentsEventsObjectplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVBComponentsEventsObjectplaceholderS"}

VBProjectsEvents Object (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproVBProjectsEventsObjectplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVBProjectsEventsObjectplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVBProjectsEventsObjectplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVBProjectsEventsObjectplaceholderS"}

Appearance Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproAppearancePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproAppearancePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproAppearancePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproAppearancePropertyActiveXControlsplaceholderS"}

BackColor, ForeColor Properties (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBackColorForeColorPropertiesActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBackColorForeColorPropertiesActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBackColorForeColorPropertiesActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBackColorForeColorPropertiesActiveXControlsplaceholderS"}

BorderStyle Constants (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBorderStyleConstantsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStyleConstantsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStyleConstantsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStyleConstantsActiveXControlsplaceholderS"}

Caption Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCaptionPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCaptionPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCaptionPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCaptionPropertyActiveXControlsplaceholderS"}

Change Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproChangeEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproChangeEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproChangeEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproChangeEventActiveXControlsplaceholderS"}

Clear Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproClearMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClearMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClearMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClearMethodActiveXControlsplaceholderS"}

Click Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproClickEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClickEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClickEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClickEventActiveXControlsplaceholderS"}

Clipboard Object Constants (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproClipboardObjectConstantsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproClipboardObjectConstantsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproClipboardObjectConstantsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproClipboardObjectConstantsActiveXControlsplaceholderS"}

Count Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproCountPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproCountPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproCountPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproCountPropertyActiveXControlsplaceholderS"}

DataObject Object (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataObjectObjectActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataObjectObjectActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataObjectObjectActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataObjectObjectActiveXControlsplaceholderS"}

DataObjectFiles Collection (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataObjectFilesCollectionActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataObjectFilesCollectionActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataObjectFilesCollectionActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataObjectFilesCollectionActiveXControlsplaceholderS"}

DataSource Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDataSourcePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDataSourcePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDataSourcePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDataSourcePropertyActiveXControlsplaceholderS"}

DblClick Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDblClickEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDblClickEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDblClickEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDblClickEventActiveXControlsplaceholderS"}

Drag-and-Drop Constants (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproDragandDropConstantsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproDragandDropConstantsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproDragandDropConstantsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproDragandDropConstantsActiveXControlsplaceholderS"}

Enabled Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproEnabledPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproEnabledPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproEnabledPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproEnabledPropertyActiveXControlsplaceholderS"}

FetchVerbs Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFetchVerbsMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFetchVerbsMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFetchVerbsMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFetchVerbsMethodActiveXControlsplaceholderS"}

Files Method (ActiveX Controls) (placeholders)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFilesMethodActiveXControlsplaceholdersC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFilesMethodActiveXControlsplaceholdersX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFilesMethodActiveXControlsplaceholdersA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFilesMethodActiveXControlsplaceholdersS"}

Font Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontPropertyActiveXControlsplaceholderS"}

FontName Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontNamePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontNamePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontNamePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontNamePropertyActiveXControlsplaceholderS"}

FontSize Property (ActiveX placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproFontSizePropertyActiveXPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproFontSizePropertyActiveXPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproFontSizePropertyActiveXPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproFontSizePropertyActiveXPlaceholderS"}

GetData Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproGetDataMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGetDataMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGetDataMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGetDataMethodActiveXControlsplaceholderS"}

GetFormat Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproGetFormatMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproGetFormatMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproGetFormatMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproGetFormatMethodActiveXControlsplaceholderS"}

HideSelection Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHideSelectionPropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHideSelectionPropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHideSelectionPropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHideSelectionPropertyActiveXControlscomplaceholderS"}

hWnd Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHWndPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHWndPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHWndPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHWndPropertyActiveXControlsplaceholderS"}

Item Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproItemMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproItemMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproItemMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproItemMethodActiveXControlsplaceholderS"}

Key Code Constants (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproKeyCodeConstantsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyCodeConstantsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyCodeConstantsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyCodeConstantsActiveXControlsplaceholderS"}

Key Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproKeyPropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyPropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyPropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyPropertyActiveXControlscomplaceholderS"}

KeyDown, KeyUp Events (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproKeyDownKeyUpEventsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyDownKeyUpEventsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyDownKeyUpEventsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyDownKeyUpEventsActiveXControlsplaceholderS"}

KeyPress Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproKeyPressEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproKeyPressEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproKeyPressEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproKeyPressEventActiveXControlsplaceholderS"}

Max, Min Properties (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMaxMinPropertiesActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMaxMinPropertiesActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMaxMinPropertiesActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMaxMinPropertiesActiveXControlsplaceholderS"}

MouseDown, MouseUp Events (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMouseDownMouseUpEventsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseDownMouseUpEventsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseDownMouseUpEventsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseDownMouseUpEventsActiveXControlsplaceholderS"}

MouseIcon Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMouseIconPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseIconPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseIconPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseIconPropertyActiveXControlsplaceholderS"}

MouseMove Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMouseMoveEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMouseMoveEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMouseMoveEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMouseMoveEventActiveXControlsplaceholderS"}

MousePointer Constants (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMousePointerConstantsActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerConstantsActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerConstantsActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerConstantsActiveXControlsplaceholderS"}

MousePointer Property (ActiveX Controls) (placeholder

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproMousePointerPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproMousePointerPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproMousePointerPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproMousePointerPropertyActiveXControlsplaceholderS"}

OLECompleteDrag Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLECompleteDragEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLECompleteDragEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLECompleteDragEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLECompleteDragEventActiveXControlsplaceholderS"}

OLEDrag Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragMethodActiveXControlsplaceholderS"}

OLEDragDrop Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragDropEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragDropEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragDropEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragDropEventActiveXControlsplaceholderS"}

OLEDragMode Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragModePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragModePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragModePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragModePropertyActiveXControlsplaceholderS"}

OLEDragOver Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDragOverEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDragOverEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDragOverEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDragOverEventActiveXControlsplaceholderS"}

OLEDropMode Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEDropModePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEDropModePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEDropModePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEDropModePropertyActiveXControlsplaceholderS"}

OLEGiveFeedback Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEGiveFeedbackEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEGiveFeedbackEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEGiveFeedbackEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEGiveFeedbackEventActiveXControlsplaceholderS"}

OLESetData Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLESetDataEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLESetDataEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLESetDataEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLESetDataEventActiveXControlsplaceholderS"}

OLEStartDrag Event (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproOLEStartDragEventActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproOLEStartDragEventActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproOLEStartDragEventActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproOLEStartDragEventActiveXControlsplaceholderS"}

Picture Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPicturePropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPicturePropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPicturePropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPicturePropertyActiveXControlsplaceholderS"}

Property Pages Dialog Box (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPropertyPagesDialogBoxActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyPagesDialogBoxActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyPagesDialogBoxActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyPagesDialogBoxActiveXControlsplaceholderS"}

RemoteHost Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRemoteHostPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemoteHostPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemoteHostPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemoteHostPropertyActiveXControlsplaceholderS"}

RemotePort Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRemotePortPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRemotePortPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRemotePortPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRemotePortPropertyActiveXControlsplaceholderS"}

SelLength, SelStart, SelText Properties (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSelLengthSelStartSelTextPropertiesActiveXControlsplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSelLengthSelStartSelTextPropertiesActiveXControlsplaceholderX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSelLengthSelStartSelTextPropertiesActiveXControlsplaceholderA"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSelLengthSelStartSelTextPropertiesActiveXControlsplaceholderS"}

SetData Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproSetDataMethodActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproSetDataMethodActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproSetDataMethodActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproSetDataMethodActiveXControlsplaceholderS"}

ShowTips Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproShowTipsPropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproShowTipsPropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproShowTipsPropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproShowTipsPropertyActiveXControlscomplaceholderS"}

Text Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTextPropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTextPropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTextPropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTextPropertyActiveXControlscomplaceholderS"}

Value Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproValuePropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproValuePropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproValuePropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproValuePropertyActiveXControlscomplaceholderS"}

hDC Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHDCPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHDCPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHDCPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHDCPropertyActiveXControlsplaceholderS"}

Height, Width Properties (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHeightWidthPropertiesActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHeightWidthPropertiesActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHeightWidthPropertiesActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHeightWidthPropertiesActiveXControlsplaceholderS"}

Index Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthIndexPropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthIndexPropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthIndexPropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthIndexPropertyActiveXControlscomplaceholderS"}

Left, Top Properties (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproLeftTopPropertiesActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproLeftTopPropertiesActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproLeftTopPropertiesActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproLeftTopPropertiesActiveXControlsplaceholderS"}

Tag Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproTagPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproTagPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproTagPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproTagPropertyActiveXControlsplaceholderS"}

Visible Property (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproVisiblePropertyActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproVisiblePropertyActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproVisiblePropertyActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproVisiblePropertyActiveXControlscomplaceholderS"}

Remove Method (ActiveX Controls) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthRemoveMethodActiveXControlscomplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthRemoveMethodActiveXControlscomplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthRemoveMethodActiveXControlscomplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthRemoveMethodActiveXControlscomplaceholderS"}

Object Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmthObjectPropertyActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbmthObjectPropertyActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbmthObjectPropertyActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbmthObjectPropertyActiveXControlsplaceholderS"}

Property Pages (ActiveX controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproPropertyPagesActiveXControlsplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproPropertyPagesActiveXControlsplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproPropertyPagesActiveXControlsplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproPropertyPagesActiveXControlsplaceholderS"}

No fonts exist (Error 24574) (Common Dialog Control) (complaceholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproNoFontsExistError24574CommonDialogControlcomplaceholderC"}
{ewc HLP95EN.DLL,DYNALINK,"Example":"vbproNoFontsExistError24574CommonDialogControlcomplaceholderX":1}
{ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproNoFontsExistError24574CommonDialogControlcomplaceholderA"}
{ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproNoFontsExistError24574CommonDialogControlcomplaceholderS"}

Help Contents placeholder

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproHelpContentsPlaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproHelpContentsPlaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproHelpContentsPlaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproHelpContentsPlaceholderS"}

BorderStyle Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproBorderStyleplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproBorderStyleplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproBorderStyleplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproBorderStyleplaceholderS"}

RightToLeft Property (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbproRightToLeftplaceholderC"} {ewc HLP95EN.DLL,DYNALINK,"Example":"vbproRightToLeftplaceholderX":1} {ewc HLP95EN.DLL,DYNALINK,"Applies To":"vbproRightToLeftplaceholderA"} {ewc HLP95EN.DLL,DYNALINK,"Specifics":"vbproRightToLeftplaceholderS"}

Refresh Method (ActiveX Controls) (placeholder)

{ewc HLP95EN.DLL,DYNALINK,"See Also":"vbmscRefreshMethodActiveXControlsplaceholderC"}