APPENDIX C

ADO 2.5 Objects and Collections

THIS APPENDIX COVERS ADO 2.5 OBJECTS and collections, and provides a complete reference on the methods, properties, and events of those objects and collections. Enumerated constants for parameter values of some methods are also provided at the end of this appendix.

Command Object

A Command object is a definition of a specific command that you intend to execute against a data source.

Methods

Cancel

Cancels execution of a pending, asynchronous `Execute` method call.

Syntax

```
command.Cancel
```

CreateParameter

Creates a new Parameter object with the specified properties.

Syntax

```
Set parameter = command.CreateParameter (Name, Type, Direction, Size, Value)
```

Return Value

Returns a Parameter object.

Parameters

Name (optional): A String representing the name of the Parameter object.

Type (optional): A Long value specifying the data type of the Parameter object. See the Type property information listed under the Parameter object section in this appendix for valid settings.

Direction (optional): A Long value specifying the type of Parameter object. See the Direction of the Parameter object for valid settings.

Size (optional): A Long value specifying the maximum length for the parameter value in characters or bytes.

Value (optional): A Variant specifying the value for the Parameter object.

Execute

Executes the query, SQL statement, or stored procedure specified in the CommandText property.

Syntax

For a row-returning Command, the syntax is

```
Set recordset = command.Execute( RecordsAffected, Parameters, Options )
```

For a non–row-returning Command, the syntax is

```
command.Execute RecordsAffected, Parameters, Options
```

Return Value

Returns a Recordset object reference.

Parameters

RecordsAffected (optional): A Long variable to which the provider returns the number of records that the operation affected.

Parameters (optional): A Variant array of parameter values passed with a SQL statement. (Output parameters will not return correct values when passed in this argument.)

Options (optional): A Long value that indicates how the provider should evaluate the CommandText property of the Command object. The value of this parameter can be one of the CommandTypeEnum constants.

Properties

ActiveConnection

Indicates to which Connection object the specified Command object currently belongs. For Command objects, this property is read/write.

Settings and Return Values

Sets or returns a String containing the definition for a connection or a Connection object. Default is a Null object reference.

CommandText

Contains the text of a command that you want to issue against a provider.

Settings and Return Values

Sets or returns a String value containing a provider command, such as an SQL statement, a table name, or a stored procedure call. Default is **“”** (zero-length string).

CommandTimeout

Indicates how long to wait while executing a command before terminating the attempt and generating an error.

Settings and Return Values

Sets or returns a Long value that indicates, in seconds, how long to wait for a command to execute. Default is 30.

CommandType

Indicates the type of a Command object.

Settings and Return Values

Sets or returns one of the `CommandTypeEnum` values.

Name

Indicates the name of the Command object.

Parameters

Contains all the Parameter objects for the Command object.

Prepared

Indicates whether or not to save a compiled version of a command before execution.

Settings and Return Values

Sets or returns a Boolean value.

Properties

Contains all the Property objects for the Command object.

State

Describes whether the Command object is open or closed.

Return Value

Returns a Long value that can be one of the `ObjectStateEnum` values.

Connection Object

A Connection object represents an open connection to a data source.

Methods

BeginTrans

Begins a new transaction.

Syntax

```
level = connection.BeginTrans()
connection.BeginTrans
```

Return Value

`BeginTrans` can be called as a function that returns a Long variable indicating the nesting level of the transaction.

Cancel

Cancels execution of a pending, asynchronous Execute or Open method call.

Syntax

```
connection.Cancel
```

Close

Closes an open Connection object and any dependent objects.

Syntax

```
connection.Close
```

CommitTrans

Saves any changes and ends the current transaction. It may also start a new transaction.

Syntax

```
connection.CommitTrans
```

Execute

Executes the specified query, SQL statement, stored procedure, or provider-specific text.

Syntax

For a non–row-returning command string, the syntax is

```
connection.Execute CommandText, RecordsAffected, Options
```

For a row-returning command string, the syntax is

```
Set recordset = connection.Execute (CommandText, RecordsAffected, Options)
```

Return Value

Returns a Recordset object reference.

Parameters

`CommandText`: A String containing the SQL statement, table name, stored procedure, or provider-specific text to execute.

`RecordsAffected` (optional): A Long variable to which the provider returns the number of records that the operation affected.

`Options` (optional): A Long value that indicates how the provider should evaluate the `CommandText` argument. Can be one or more `CommandTypeEnum` or `ExecuteOptionEnum` values.

Open

Opens a connection to a data source.

Syntax

```
connection.Open ConnectionString, UserID, Password, OpenOptions
```

Parameters

`ConnectionString` (optional): A String containing connection information. See the `ConnectionString` property information later in this appendix for details on valid settings.

`UserID` (optional): A String containing a user name to use when establishing the connection.

`Password` (optional): A String containing a password to use when establishing the connection.

`OpenOptions` (optional): A `ConnectOptionEnum` value. If set to `adConnectAsync`, the connection will be opened asynchronously. A `ConnectComplete` event will be issued when the connection is available.

OpenSchema

Obtains database schema information from the provider.

Syntax

```
Set recordset = connection.OpenSchema (QueryType, Criteria, SchemaID)
```

Return Values

Returns a Recordset object that contains schema information. The Recordset will be opened as a read-only, static cursor.

Parameters

`QueryType`: The type of schema query to run. Can be any of a SchemaEnum value.

Criteria (optional): An array of query constraints for each QueryType option that can be any of a SchemaEnum value.

SchemaID: The GUID for a provider-schema schema query not defined by the OLE DB specification. This parameter is required if QueryType is set to adSchemaProviderSpecific; otherwise, it is not used.

RollbackTrans

Cancels any changes made during the current transaction and ends the transaction. It may also start a new transaction.

Syntax

connection.**RollbackTrans**

Properties

Attributes

Indicates one or more characteristics of a Connection object.

Settings and Return Values

Sets or returns a Long value. The property is read/write, and its value can be the sum of any one or more of XactAttributeEnum values (default is zero).

CommandTimeout

Indicates how long to wait while executing a command before terminating the attempt and generating an error.

Settings and Return Values

Sets or returns a Long value that indicates, in seconds, how long to wait for a command to execute. Default is 30.

ConnectionString

Contains the information used to establish a connection to a data source.

Settings and Return Values

Sets or returns a String value. ADO supports four arguments for the `ConnectionString` property; any other arguments pass directly to the provider without any processing by ADO. The arguments that ADO supports are listed in Table C-1.

Table C-1. Arguments Supported by ADO

ARGUMENT	DESCRIPTION
`Provider=`	Specifies the name of a provider to use for the connection.
`File Name=`	Specifies the name of a provider-specific file (for example, a persisted data source object) containing preset connection information.
`Remote Provider=`	Specifies the name of a provider to use when opening a client-side connection. (Remote Data Service only.)
`Remote Server=`	Specifies the path name of the server to use when opening a client-side connection. (Remote Data Service only.)
`URL=`	Specifies the connection string as an absolute URL identifying a resource, such as a file or directory.

ConnectionTimeout

Indicates how long to wait while establishing a connection before terminating the attempt and generating an error.

Settings and Return Values

Sets or returns a Long value that indicates, in seconds, how long to wait for the connection to open. Default is 15.

CursorLocation

Sets or returns the location of the cursor engine.

Settings and Return Values

Sets or returns a Long value that can be set to one of the `CursorLocationEnum` constants.

DefaultDatabase

Indicates the default database for the Connection object.

Settings and Return Values

Sets or returns a String that evaluates to the name of a database available from the provider.

Errors

Contains all the Error objects created in response to a single failure involving the provider.

IsolationLevel

Indicates the level of isolation for the Connection object.

Settings and Return Values

Sets or returns one of the `IsolationLevelEnum` values.

Mode

Indicates the available permissions for modifying data in the Connection object.

Settings and Return Values

Sets or returns one of the `ConnectModeEnum` values.

Properties

Contains all the Property objects for the Connection object.

Provider

Indicates the name of the provider for the Connection object.

Settings and Return Values

Sets or returns a String value.

State

Describes whether the state of the Connection object is open or closed.

Return Value

Returns a Long value that can be one of the `ObjectStateEnum` constants.

Version

Indicates the ADO version number.

Return Value

Returns a String value.

Events

BeginTransComplete

This event is fired after the `BeginTrans` operation.

Syntax

```
BeginTransComplete TransactionLevel, pError, adStatus, pConnection
```

Parameters

`TransactionLevel`: A Long. Contains the new transaction level of the BeginTrans that caused this event.

`pError`: An Error object. It describes the error that occurred if the value of `EventStatusEnum` is `adStatusErrorsOccurred`; otherwise it is not set.

adStatus: An `EventStatusEnum` status value. When any of these methods is called, this parameter is set to `adStatusOK` if the operation that caused the event was successful, or `adStatusErrorsOccurred` if the operation failed.

pConnection: The Connection object for which this event occurred.

CommitTransComplete

This event is fired after the `CommitTrans` operation.

Syntax

```
CommitTransComplete pError, adStatus, pConnection
```

Parameters

pError: An Error object. It describes the error that occurred if the value of `EventStatusEnum` is `adStatusErrorsOccurred`; otherwise it is not set.

adStatus: An `EventStatusEnum` status value. When any of these methods is called, this parameter is set to `adStatusOK` if the operation that caused the event was successful, or `adStatusErrorsOccurred` if the operation failed.

pConnection: The Connection object for which this event occurred.

ConnectComplete

This event is fired after a connection starts.

Syntax

```
ConnectComplete pError, adStatus, pConnection
```

Parameters

pError: An Error object. It describes the error that occurred if the value of *adStatus* is `adStatusErrorsOccurred`; otherwise it is not set.

adStatus: An `EventStatusEnum` status value. When either of these methods is called, this parameter is set to `adStatusOK` if the operation that caused the event was successful, or `adStatusErrorsOccurred` if the operation failed.

pConnection: The Connection object for which this event applies.

Disconnect

This event is fired after the connection ends.

Syntax

```
Disconnect adStatus, pConnection
```

Parameters

adStatus: An EventStatusEnum status value. When either of these methods is called, this parameter is set to adStatusOK if the operation that caused the event was successful, or adStatusErrorsOccurred if the operation failed.

pConnection: The Connection object for which this event applies.

ExecuteComplete

This event is fired after a command has finished executing.

Syntax

```
ExecuteComplete RecordsAffected, pError, adStatus, pCommand, pRecordset,
pConnection
```

Parameters

RecordsAffected: A Long. The number of records affected by the command.

pError: An Error object. It describes the error that occurred if the value of *adStatus* is adStatusErrorsOccurred; otherwise it is not set.

adStatus: An EventStatusEnum status value. When this method is called, this parameter is set to adStatusOK if the operation that caused the event was successful, or adStatusErrorsOccurred if the operation failed.

pCommand: The Command object, if any, that was executed.

pRecordset: A Recordset object. The result of the execution. This recordset may be empty.

pConnection: A Connection object. The connection on which the command was executed.

InfoMessage

This event is fired whenever a ConnectionEvent operation completes successfully and additional information is returned by a provider.

Syntax

```
InfoMessage pError, adStatus, pConnection
```

Parameters

pError: An Error object. It describes the error that occurred if the value of *adStatus* is adStatusErrorsOccurred; otherwise it is not set. Multiple warnings can be returned, which can be found by enumerating the Errors collection.

adStatus: An EventStatusEnum status value. When this method is called, this parameter is set to adStatusOK if the operation that caused the event was successful, or adStatusErrorsOccurred if the operation failed.

pConnection: A Connection object. The connection on which the command was executed.

RollbackTransComplete

This event is fired after the RollbackTrans operation.

Syntax

```
RollbackTransComplete pError, adStatus, pConnection
```

Parameters

pError: An Error object. It describes the error that occurred if the value of EventStatusEnum is adStatusErrorsOccurred; otherwise it is not set.

adStatus: An EventStatusEnum status value. When any of these methods is called, this parameter is set to adStatusOK if the operation that caused the event was successful, or adStatusErrorsOccurred if the operation failed.

pConnection: The Connection object for which this event occurred.

WillConnect

This event is fired before a connection starts. The parameters to be used in the pending connection are supplied as input parameters and can be changed before the method returns. This event may return a request that the pending connection be canceled.

Syntax

```
WillConnect ConnectionString, UserID, Password, Options, adStatus, pConnection
```

Parameters

ConnectionString: A String containing connection information for the pending connection.

UserID: A String containing a user name for the pending connection.

Password: A String containing a password for the pending connection.

Options: A Long value that indicates how the provider should evaluate the ConnectionString. See the CommandType property description in this appendix for a list of acceptable values.

adStatus: An EventStatusEnum status value.

pConnection: The Connection object for which this event notification applies.

WillExecute

This event is fired just before a pending command executes on the connection and affords the user an opportunity to examine and modify the pending execution parameters. This event may return a request that the pending command be canceled.

Syntax

```
WillExecute Source, CursorType, LockType, Options, adStatus, pCommand, pRecordset,
pConnection
```

Parameters

Source: A String containing an SQL command or a stored procedure name.

CursorType: A `CursorTypeEnum` containing the type of cursor for the recordset that will be opened. This parameter cannot be changed if it is set to `adOpenUnspecified` when this method is called.

LockType: A `LockTypeEnum` containing the lock type for the recordset that will be opened. This parameter cannot be changed if it is set to `adLockUnspecified` when this method is called.

Options: A Long value of options that can be used to execute the command or open the recordset.

adStatus: An `EventStatusEnum` status value that may be `adStatusCantDeny` or `adStatusOK` when this method is called. If it is `adStatusCantDeny`, this method may not request cancellation of the pending operation.

pCommand: The Command object for which this event notification applies.

pRecordset: The Recordset object for which this event notification applies.

pConnection: The Connection object for which this event notification applies.

Error Object

An Error object contains details about data access errors pertaining to a single operation involving the provider.

Properties

Description

A descriptive string associated with an Error object.

Return Value

Returns a String value.

HelpContext

Indicates the ContextID in the help file for the associated error.

Return Value

Returns an Integer.

HelpFile

Indicates the name of the help file.

Return Value

Returns a String.

NativeError

Indicates the provider-specific error code for a given Error object.

Return Value

Returns a Long value.

Number

Indicates the number that uniquely identifies an Error object.

Return Value

Returns a Long value.

Source

Indicates the name of the object or application that originally generated an error.

Return Value

Returns a String value.

SQLState

Indicates the SQL state for a given Error object.

Return Value

Returns a five-character String that follows the ANSI SQL standard.

Errors Collection

The Errors collection contains all the Error objects created in response to a single failure involving the provider.

Methods

Clear

Removes all Error objects in the collection.

Syntax

```
errors.Clear
```

Refresh

Updates the Error objects in the collection to reflect Error objects available from, and specific to, the provider.

Syntax

```
errors.Refresh
```

Properties

Count

Indicates the number of Error objects in the collection.

Return Value

Returns a Long value.

Item

Allows indexing into the Errors collection to reference a specific Error object.

Field Object

A Field object represents a column of data with a common data type.

Methods

AppendChunk

Appends data to a large text or binary data Field object.

Syntax

```
field.AppendChunk Data
```

Parameter

Data: A Variant containing the data to be appended to the Field object.

GetChunk

Returns all or a portion of the contents of a large text or binary data Field object.

Syntax

```
variable = field.GetChunk( Size )
```

Return Value

Returns a Variant.

Parameter

`Size`: A Long expression equal to the number of bytes or characters to be retrieved.

Properties

ActualSize

Indicates the actual length of a field's value.

Settings and Return Values

Returns a Long value. Some providers may allow this property to be set to reserve space for BLOB data, in which case the default value is 0.

Attributes

Indicates characteristics of the Field object.

Settings and Return Values

Sets or returns a Long value.

The `Attributes` property is read-only, and its value can be the sum of any one or more of the `FieldAttributeEnum` values.

DataFormat

Identifies the format that the data should be displayed in.

Return Value

Returns a Variant value.

DefinedSize

Indicates the defined size of the Field object.

Return Value

Returns a Long value that reflects the defined size of a field as a number of bytes.

Name

Indicates the name of the Field object.

Return Value

Returns a String value. The value is read-only.

NumericScale

Indicates the scale of numeric values in the Field object.

Setting

Sets a Byte value, indicating the number of decimal places to which numeric values will be resolved.

OriginalValue

Indicates the value of the Field object that existed in the record before any changes were made.

Return Value

Returns a Variant value.

Precision

Indicates the degree of precision for numeric the Field object.

Return Value

Sets or returns a Byte value, indicating the maximum total number of digits used to represent values. The value is read-only.

Properties

Contains all the Property objects for the Field object.

Type

Indicates the operational type or data type of the Field object.

Settings and Return Values

Sets or returns one of the `DataTypeEnum` values.

UnderlyingValue

Indicates the Field object's current value in the database.

Return Value

Returns a Variant value.

Value

Indicates the value assigned to the Field object.

Settings and Return Values

Sets or returns a Variant value. Default value depends on the `Type` property.

Fields Collection

A Fields collection contains all the Field objects of a Recordset object.

Methods

Append

Appends a Field object to the Fields collection. A new Field object may be created before it is appended to the collection.

Syntax

```
fields.Append Name, Type, DefinedSize, Attrib
```

Parameters

Name: A String. The name of the new Field object, which must not be the same name as any other object in *fields*.

Type: A DataTypeEnum, whose default value is adEmpty. The data type of the new field.

DefinedSize (optional): A Long. The defined size in characters or bytes of the new field. The default value for this parameter is derived from *Type*. (The default *Type* is adEmpty, and default *DefinedSize* is unspecified.)

Attrib (optional): A FieldAttributeEnum, whose default value is adFldDefault. Specifies attributes for the new field. If this value is not specified, the field will contain attributes derived from Type.

CancelUpdate

Cancels any changes made to the Fields collection.

Syntax

```
fields.CancelUpdate
```

Delete

Deletes a Field object from the collection.

Syntax

```
fields.Delete Field
```

Parameter

`Field`: A Variant designating the Field object to delete. This parameter must be the name of the Field object; it cannot be an ordinal position or the Field object itself.

Refresh

Updates the Field objects in the collection to reflect Field objects available from and specific to the provider.

Syntax

```
fields.Refresh
```

Resync

Resynchronizes the data in the Field objects.

Syntax

```
fields.Resync
```

Update

Saves any changes to the collection.

Syntax

```
fields.Update
```

Properties

Count

Indicates the number of objects in the collection.

Return Value

Returns a Long value.

Item

Allows indexing into the Fields collection to reference a specific Field object.

Parameter Object

A Parameter object represents a parameter or argument associated with a Command object based on a parameterized query or stored procedure.

Method

AppendChunk

Appends data to a large text or binary data Parameter object.

Syntax

```
parameter.AppendChunk Data
```

Parameter

Data: A Variant containing the data to be appended to the Parameter object.

Properties

Attributes

Indicates characteristics of the Parameter object.

Settings and Return Values

Sets or returns a Long value. The value can be the sum of any one or more of the ParameterAttributesEnum values.

Direction

Indicates whether the Parameter object represents an input parameter, an output parameter, or both, or if the parameter is the return value from a stored procedure.

Settings and Return Values

Sets or returns one of the `ParameterDirectionEnum` values.

Name

Indicates the name of the Parameter object.

Return Value

Returns a String.

NumericScale

Indicates the scale of numeric values in the Parameter object.

Settings and Return Values

Sets or returns a Byte value, indicating the number of decimal places to which numeric values will be resolved.

Precision

Indicates the degree of precision for numeric values in the Parameter object.

Settings and Return Values

Sets or returns a Byte value, indicating the maximum total number of digits used to represent values.

Properties

Contains all the Property objects in the Parameter object.

Size

Indicates the maximum size, in bytes or characters, of the Parameter object.

Settings and Return Values

Sets or returns a Long value that indicates the maximum size in bytes or characters of a value in the Parameter object.

Type

Indicates the operational type or data type of the Parameter object.

Settings and Return Values

Sets or returns one of the `DataTypeEnum` values.

Value

Indicates the value assigned to the Parameter object.

Settings and Return Values

Sets or returns a Variant value. Default value depends on the Type property.

Parameters Collection

A Parameters collection contains all the Parameter objects of a Command object.

Methods

Append

Appends a Parameter object to the collection.

Syntax

parameters.Append *parameter*

Parameter

`parameter`: A Parameter object variable representing the Parameter object to be appended.

Delete

Deletes a Parameter object from the collection.

Syntax

```
parameters.Delete Index
```

Parameter

`Index`: A String representing the name of the object you want to delete, or the object's ordinal position (index) in the collection.

Refresh

Updates the Parameter objects in the collection to reflect Parameter objects available from, and specific to, the provider.

Syntax

```
parameters.Refresh
```

Properties

Count

Indicates the number of Parameter objects in the collection.

Return Value

Returns a Long value.

Item

Allows indexing into the Parameters collections to reference a specific Parameter object.

Properties Collection

A Properties collection contains all the Property objects for a specific instance of an object.

Method

Refresh

Updates the Property objects in the collection to reflect Property objects available from and specific to the provider.

Syntax

```
properties.Refresh
```

Properties

Count

Indicates the number of Property objects in the collection.

Return Value

Returns a Long value.

Item

Allows indexing into the Properties collection to reference a specific Property object.

Property Object

A Property object represents a dynamic characteristic of an ADO object that is defined by the provider.

Properties

Attributes

Indicates characteristics of a Property object.

Settings and Return Values

Sets or returns a Long value. The value can be the sum of any one or more of the `PropertyAttributesEnum` values.

Name

Indicates the name of the Property object.

Return Value

Returns a String value.

Type

Indicates the operational type or data type of a Property object.

Settings and Return Values

Sets or returns one of the `DataTypeEnum` values.

Value

Indicates the value assigned to the Property object.

Settings and Return Values

Sets or returns a Variant value. Default value depends on the Type property.

Record Object

Represents a row in a Recordset, or a file or directory in a file system.

Methods

Cancel

Cancels execution of a pending, asynchronous method call.

Syntax

```
record.Cancel
```

Close

Closes an open Record object and any dependent objects.

Syntax

```
record.Close
```

CopyRecord

Copies a file or directory, and its contents, to another location.

Syntax

```
record.CopyRecord (Source, Destination, UserName, Password, Options, Async)
```

Parameters

Source (optional): A String value that contains a URL specifying the file or directory to be copied. If *Source* is omitted or specifies an empty string, the file or directory represented by this Record will be copied.

Destination (optional): A String value that contains a URL specifying the location where *Source* will be copied.

UserName (optional): A String value that contains the user ID that, if needed, authorizes access to *Destination*.

Password (optional): A String value that contains the password that, if needed, verifies *UserName*.

Options (optional): A CopyRecordOptionsEnum value that has a default value of adCopyUnspecified. Specifies the behavior of this method.

Async (optional): A Boolean value that when True specifies that this operation should be asynchronous.

Return Value

A String value that typically returns the value of Destination. However, the exact value returned is provider-dependent.

DeleteRecord

Deletes a file or directory, and all of its subdirectories.

Syntax

```
record.DeleteRecord Source, Async
```

Parameters

Source (optional): A String value that contains a URL identifying the file or directory to be deleted. If *Source* is omitted or specifies an empty string, the file or directory represented by this Record is deleted.

Async (optional): A Boolean value that, when True, specifies that the delete operation is asynchronous.

GetChildren

Returns a Recordset whose rows represent the files and subdirectories in the directory represented by this Record.

Syntax

```
Set recordset = record.GetChildren
```

Return Value

A Recordset object for which each row represents a file or directory.

MoveRecord

Moves a file, or a directory and its contents, to another location.

Syntax

```
record.MoveRecord (Source, Destination, UserName, Password, Options, Async)
```

Parameters

Source (optional): A String value that contains a URL identifying the Record to be moved. If *Source* is omitted or specifies an empty string, the file or directory represented by this Record is moved.

Destination (optional): A String value that contains a URL specifying the location where *Source* will be moved.

UserName (optional): A String value that contains the user ID that, if needed, authorizes access to *Destination*.

Password (optional): A String that contains the password that, if needed, verifies *UserName*.

Options (optional): A MoveRecordOptionsEnum value whose default value is adMoveUnspecified. Specifies the behavior of this method.

Async (optional): A Boolean value that, when True, specifies this operation should be asynchronous.

Return Value

A String value. Typically, the value of Destination is returned. However, the exact value returned is provider-dependent.

Open

Opens an existing Record object, or creates a new file or directory.

Syntax

```
record.Open Source, ActiveConnection, Mode, CreateOptions, Options, UserName,
Password
```

Parameters

Source (optional): A Variant that represents the URL of the entity to be represented by this Record object, or a row of an open Recordset object.

ActiveConnection (optional): A Variant that represents the connect string or open Connection object that specifies the file or directories (that is, the context) over which subsequent Record operations apply.

Mode (optional): A `ConnectModeEnum` value, whose default value is `adModeUnknown`, that specifies the access mode for the resultant Record object.

CreateOptions (optional): A `RecordCreateOptionsEnum` value, whose default value is `adFailIfNotExists`, that specifies whether an existing file or directory should be opened, or a new file or directory should be created. If set to the default value, the access mode is obtained from the `Mode` property.

Options (optional): A `RecordOpenOptionsEnum` value, whose default value is `adOpenRecordUnspecified`, that specifies options for opening the Record. These values may be combined.

UserName (optional): A `String` value that contains the user ID which, if needed, authorizes access to *Source*.

Password (optional): A `String` value that contains the password that, if needed, verifies *UserName*.

Properties

ActiveConnection

Indicates to which Connection object the specified Record object currently belongs.

Settings and Return Values

Sets or returns a `String` value that contains a definition for a connection if the connection is closed, or a `Variant` containing the current Connection object if the connection is open. Default is a null object reference.

Fields

Contains all the Field objects for the current Record object.

Mode

Indicates the available permissions for modifying data in the Record object.

Settings and Return Values

Sets or returns a `ConnectModeEnum` value. The default value is `adReadOnly`.

ParentURL

Indicates an absolute URL string that points to the parent Record of the current Record object.

Return Value

Returns a String value that indicates the URL of the parent Record.

Properties

Contains all the Property objects for the current Record object.

RecordType

Indicates the type of the Record object.

Return Value

Returns a `RecordTypeEnum` value.

Source

Indicates the entity represented by the Record object.

Settings and Return Values

Sets or returns a Variant value that indicates the entity represented by the Record.

State

Indicates whether the current state of the Record object is connecting, executing, or retrieving.

Return Value

Returns a Long value that can be any of the `ObjectStateEnum` value. The default value is `adStateClosed`.

Recordset Object

A Recordset object represents the entire set of records from a base table or the results of an executed command. At any time, the Recordset object refers to only a single record within the set as the current record.

Methods

AddNew

Creates a new record for an updatable Recordset object.

Syntax

```
recordset.AddNew FieldList, Values
```

Parameters

`FieldList` (optional): A single name or an array of names or ordinal positions of the fields in the new record.

`Values` (optional): A single value or an array of values for the fields in the new record. If *`FieldList`* is an array, *`Values`* must also be an array with the

same number of members; otherwise, an error occurs. The order of field names must match the order of field values in each array.

Cancel

Cancels execution of a pending, asynchronous `Execute` or `Open` method call.

Syntax

```
recordset.Cancel
```

CancelBatch

Cancels a pending batch update.

Syntax

```
recordset.CancelBatch AffectRecords
```

Parameter

`AffectRecords` (optional): An `AffectEnum` value that determines how many records the `CancelBatch` method will affect.

CancelUpdate

Cancels any changes made to the current record or to a new record prior to calling the Update method.

Syntax

```
recordset.CancelUpdate
```

Clone

Creates a duplicate Recordset object from an existing Recordset object. Optionally, specifies that the clone be read-only.

Syntax

```
Set rstDuplicate = rstOriginal.Clone (LockType)
```

Return Value

Returns a Recordset object reference.

Parameters

`rstDuplicate`: An object variable identifying the duplicate Recordset object you're creating.

`rstOriginal`: An object variable identifying the Recordset object you want to duplicate.

`LockType` (optional): A `LockTypeEnum` value that specifies either the lock type of the original Recordset, or a read-only Recordset.

Close

Closes the Recordset object.

Syntax

```
recordset.Close
```

CompareBookmarks

Compares two bookmarks and returns an indication of their relative values.

Syntax

```
result = recordset.CompareBookmarks(Bookmark1, Bookmark2) As CompareEnum
```

Return Value

Returns a `CompareEnum` value that indicates the relative row position of two records represented by their bookmarks.

Parameters

Bookmark1: The bookmark of the first row.

Bookmark2: The bookmark of the second row.

Delete

Deletes the current record or a group of records.

Syntax

```
recordset.Delete AffectRecords
```

Parameter

AffectRecords: An `AffectEnum` value that determines how many records the Delete method will affect.

Find

Searches a Recordset for the record that satisfies the specified criteria. If the criteria is met, the recordset position is set on the found record; otherwise, the position is set on the end of the recordset.

Syntax

```
recordset.Find (criteria, SkipRows, searchDirection, start)
```

Parameters

criteria: A String containing a statement that specifies the column name, comparison operator, and value to use in the search.

SkipRows: An optional Long value, whose default value is zero, that specifies the offset from the current row or *start* bookmark to begin the search.

searchDirection: An optional `SearchDirectionEnum` value that specifies whether the search should begin on the current row or the next available row in the direction of the search. Its value can be `adSearchForward` or

adSearchBackward. The search stops at the start or end of the recordset, depending on the value of *searchDirection*.

start: An optional Variant bookmark to use as the starting position for the search.

GetRows

Retrieves multiple records of a Recordset into an array.

Syntax

```
array = recordset.GetRows( Rows, Start, Fields )
```

Return Values

Returns a two-dimensional array.

Parameters

Rows (optional): A Long expression indicating the number of records to retrieve. Default is adGetRowsRest (1).

Start (optional): A String or Variant that evaluates to the bookmark for the record from which the GetRows operation should begin. You can also use one of the BookmarkEnum values.

Fields (optional): A Variant representing a single field name or ordinal position or an array of field names or ordinal position numbers. ADO returns only the data in these fields.

GetString

Returns the Recordset as a string.

Syntax

```
Set Variant = recordset.GetString(StringFormat, NumRows, ColumnDelimiter,
RowDelimiter, NullExpr)
String = recordset.GetString(StringFormat, NumRows, ColumnDelimiter, RowDelimiter,
NullExpr)
```

Return Value

Returns the Recordset as a string-valued Variant (BSTR).

Parameters

StringFormat: A StringFormatEnum value that specifies how the Recordset should be converted to a string. The *RowDelimiter*, *ColumnDelimiter*, and *NullExpr* parameters are used only with a *StringFormat* of adClipString.

NumRows (optional): The number of rows in the recordset to convert. If *NumRows* is not specified, or if it is greater than the total number of rows in the recordset, then all the rows in the recordset are converted.

ColumnDelimiter (optional): Delimiter used between columns if specified, otherwise the TAB character.

RowDelimiter (optional): Delimiter used between rows if specified, otherwise the CARRIAGE RETURN character.

NullExpr (optional): Expression used in place of a NULL value if specified, otherwise the empty string.

Move

Moves the position of the current record in a Recordset object.

Syntax

```
recordset.Move NumRecords, Start
```

Parameters

NumRecords: A signed Long expression specifying the number of records the current record position moves.

Start (optional): A String or Variant that evaluates to a bookmark. You can also use one of the BookmarkEnum values.

MoveFirst

Moves to the first record in a specified Recordset object and makes that record the current record.

Syntax

```
recordset.MoveFirst
```

MoveLast

Moves to the last record in a specified Recordset object and makes that record the current record.

Syntax

```
recordset.MoveLast
```

MoveNext

Moves to the next record in a specified Recordset object and makes that record the current record.

Syntax

```
recordset. MoveNext
```

MovePrevious

Moves to the previous record in a specified Recordset object and makes that record the current record.

Syntax

```
recordset.MovePrevious
```

NextRecordset

Clears the current Recordset object and returns the next Recordset by advancing through a series of commands.

Syntax

```
Set recordset2 = recordset1.NextRecordset( RecordsAffected )
```

Return Value

Returns a Recordset object. In the syntax model, *recordset1* and *recordset2* can be the same Recordset object, or separate objects can be used.

Parameter

RecordsAffected (optional): A Long variable to which the provider returns the number of records that the current operation affected.

Open

Opens a cursor.

Syntax

```
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

Parameters

Source (optional): A Variant that evaluates to a valid Command object variable name, an SQL statement, a table name, a stored procedure call, or the file name of a persisted Recordset.

ActiveConnection (optional): Either a Variant that evaluates to a valid Connection object variable name, or a String containing `ConnectionString` parameters.

CursorType (optional): A `CursorTypeEnum` value that determines the type of cursor that the provider should use when opening the Recordset.

LockType (optional): A `LockTypeEnum` value that determines which type of locking (concurrency) the provider should use when opening the Recordset.

Options (optional): A Long value that indicates how the provider should evaluate the *Source* argument if it represents something other than a Command object, or that the Recordset should be restored from a file where it was previously saved. Can be one or more CommandTypeEnum or ExecuteOptionEnum values.

Requery

Updates the data in a Recordset object by reexecuting the query on which the object is based.

Syntax

```
recordset.Requery Options
```

Parameter

Options (optional): A bitmask indicating options affecting this operation. If this parameter is set to adExecuteAsync, this operation will execute asynchronously and a RecordChangeComplete event will be issued when it concludes.

Resync

Refreshes the data in the current Recordset object from the underlying database.

Syntax

```
recordset.Resync AffectRecords, ResyncValues
```

Parameters

AffectRecords (optional): An AffectEnum value that determines how many records the Resync method will affect.

ResyncValues (optional): A ResyncEnum value that specifies whether underlying values are overwritten.

Save

Saves (persists) the Recordset in a file.

Syntax

```
recordset.Save FileName, PersistFormat
```

Parameters

FileName (optional): Complete path name of the file where the Recordset is to be saved.

PersistFormat (optional): The format in which the Recordset is to be saved. Currently the default, and only, valid value is adPersistADTG.

Seek

Searches the index of a Recordset to quickly locate the row that matches the specified values, and changes the current row position to that row.

Syntax

```
recordset.Seek KeyValues, SeekOption
```

Parameters

KeyValues: An array of Variant values. An index consists of one or more columns and the array contains a value to compare against each corresponding column.

SeekOption: A SeekEnum value that specifies the type of comparison to be made between the columns of the index and the corresponding *KeyValues*.

Supports

Determines whether a specified Recordset object supports a particular type of functionality.

Syntax

```
boolean = recordset.Supports( CursorOptions )
```

Return Value

Returns a `Boolean` value that indicates whether all of the features identified by the *`CursorOptions`* argument are supported by the provider.

Parameter

`CursorOptions`: A `Long` expression that consists of one or more of the `CursorOptionEnum` values.

`Update`

Saves any changes you make to the current record of a Recordset object.

Syntax

```
recordset.Update Fields, Values
```

Parameters

`Fields` (optional): A Variant representing a single name or a Variant array representing names or ordinal positions of the field or fields you wish to modify.

`Values` (optional): A Variant representing a single value or a Variant array representing values for the field or fields in the new record.

`UpdateBatch`

Writes all pending batch updates to disk.

Syntax

```
recordset.UpdateBatch AffectRecords
```

Parameter

`AffectRecords` (optional): An `AffectEnum` value that determines how many records the `UpdateBatch` method will affect.

Properties

AbsolutePage

Specifies in which page the current record resides.

Settings and Return Values

Sets or returns a Long value from 1 to the number of pages in the Recordset object (`PageCount`), or returns one of the `PositionEnum` values.

AbsolutePosition

Specifies the ordinal position of a Recordset object's current record.

Settings and Return Values

Sets or returns a Long value from 1 to the number of records in the Recordset object (`RecordCount`), or returns one of the `PositionEnum` values.

ActiveCommand

Indicates the Command object that created the associated Recordset object.

Return Value

Returns a Variant containing a Command object. Default is a Null object reference.

ActiveConnection

Indicates to which Connection object the specified Recordset object currently belongs.

Settings and Return Values

Sets or returns a String containing the definition for a connection or a Connection object. Default is a Null object reference.

BOF

Indicates that the current record position is before the first record in a Recordset object.

Return Value

Returns Boolean values.

Bookmark

Returns a bookmark that uniquely identifies the current record in a Recordset object or sets the current record in a Recordset object to the record identified by a valid bookmark.

Settings and Return Values

Sets or returns a Variant expression that evaluates to a valid bookmark.

CacheSize

Indicates the number of records from a Recordset object that are cached locally in memory.

Settings and Return Values

Sets or returns a Long value that must be greater than 0. Default is 1.

CursorLocation

Sets or returns the location of the cursor engine.

Settings And Return Values

Sets or returns a Long value that can be set to one of the `CursorLocationEnum` values.

CursorType

Indicates the type of cursor used in a Recordset object.

Settings and Return Values

Sets or returns one of the `CursorTypeEnum` values.

DataMember

The name of the data member to retrieve from the object referenced by the DataSource property.

Settings and Return Values

Sets or returns a String value. The name is not case sensitive.

DataSource

Specifies an object containing data to be represented as a Recordset object.

EditMode

Indicates the editing status of the current record.

Return Value

Returns one of the `EditModeEnum` values.

EOF

Indicates that the current record position is after the last record in a Recordset object.

Return Value

Returns Boolean values.

Fields

Contains all the Field objects for the current Recordset object.

Filter

Indicates a filter for data in a Recordset.

Settings and Return Values

Sets or returns a Variant value, which can contain any one of the following:

- Criteria string: A string made up of one or more individual clauses concatenated with `AND` or `OR` operators.
- Array of bookmarks: An array of unique bookmark values that point to records in the Recordset object.
- One of the `FilterGroupEnum` values.

Index

Indicates a filter for data in a Recordset.

Settings and Return Values

Sets or returns a Variant value, which can contain any one of the following:

- Criteria string: A string made up of one or more individual clauses concatenated with AND or OR operators.
- Array of bookmarks: An array of unique bookmark values that point to records in the Recordset object.
- One of the FilterGroupEnum values.

LockType

Indicates the type of locks placed on records during editing.

Settings and Return Values

Sets or returns one of the `LockTypeEnum` values.

MarshalOptions

Indicates which records are to be marshaled back to the server.

Settings And Return Values

Sets or returns a Long value that can be one of the `MarshalOptionsEnum` values.

MaxRecords

Indicates the maximum number of records to return to a Recordset from a query.

Settings and Return Values

Sets or returns a Long value. Default is zero (no limit).

PageCount

Indicates how many pages of data the Recordset object contains.

Return Value

Returns a Long value.

PageSize

Indicates how many records constitute one page in the Recordset.

Settings and Return Values

Sets or returns a Long value, indicating how many records are on a page. Default is 10.

Properties

Contains all the Property objects in the current Recordset object.

RecordCount

Indicates the current number of records in a Recordset object.

Return Value

Returns a Long value.

Sort

Specifies one or more field names the Recordset is sorted on, and whether each field is sorted in ascending or descending order.

Settings and Return Values

Sets or returns a String of comma-separated field names to sort on, where each name is a Field in the Recordset, and is optionally followed by a blank and the keyword `ASCENDING` or `DESCENDING`, which specifies the field sort order.

Source

Indicates the source for the data in a Recordset object (Command object, SQL statement, table name, or stored procedure).

Settings and Return Values

Sets a String value or Command object reference; returns only a String value.

State

Describes for a Recordset object executing an asynchronous method, whether the current state of the object is connecting, executing, or fetching.

Return Value

Returns a `Long ObjectStateEnum` value. The default value is `adStateClosed`.

Status

Indicates the status of the current record with respect to batch updates or other bulk operations.

Return Value

Returns a sum of one or more of the `RecordStatusEnum` values.

StayInSync

Indicates, in a hierarchical Recordset object, whether the parent row should change when the set of underlying child records (that is, a *chapter*) changes.

Settings and Return Values

Sets or returns a Boolean value. If set to True, the parent Recordset object will be updated if the chapter changes; if False, the parent Recordset object will continue to refer to the previous chapter.

Events

EndOfRecordset

This event is fired when there is an attempt to move to a row past the end of the Recordset.

Syntax

```
EndOfRecordset fMoreData, adStatus, pRecordset
```

Parameters

`fMoreData`: A `VARIANT_BOOL`. It is possible to append new records to *`pRecordset`* while processing this event.

adStatus: An `EventStatusEnum` status value.

pRecordset: The Recordset for which this event occurred.

FetchComplete

This event is fired after all the records in a lengthy asynchronous operation have been retrieved (fetched) into the Recordset.

Syntax

```
FetchComplete pError, adStatus, pRecordset
```

Parameters

pError: An Error object. It describes the error that occurred if the value of *adStatus* is `adStatusErrorsOccurred`; otherwise it is not set.

adStatus: An `EventStatusEnum` status value. When this method is called, this parameter is set to `adStatusOK` if the operation that caused the event was successful, or `adStatusErrorsOccurred` if the operation failed.

pRecordset: The Recordset object for which the records were retrieved.

FetchProgress

This event is fired periodically during a lengthy asynchronous operation to report how many rows have currently been retrieved (fetched) into the Recordset.

Syntax

```
FetchProgress Progress, MaxProgress, pRecordset
```

Parameters

Progress: A Long. The number of records that have currently been retrieved.

MaxProgress: A Long. The maximum number of records expected to be retrieved.

pRecordset: The Recordset object for which the records are being retrieved.

FieldChangeComplete

This event is fired after the value of one or more Field objects has changed.

Syntax

```
FieldChangeComplete cFields, Fields, pError, adStatus, pRecordset
```

Parameters

cFields: A Long. The number of Field objects in *Fields*.

Fields: An array of Variants. Contains Field objects with pending changes.

pError: An Error object. It describes the error that occurred if the value of *adStatus* is adStatusErrorsOccurred; otherwise it is not set.

adStatus: An EventStatusEnum status value.

pRecordset: The Recordset object for which this event occurred.

MoveComplete

This event is fired after the current position in the Recordset changes.

Syntax

```
MoveComplete adReason, pError, adStatus, pRecordset
```

Parameters

adReason: An EventReasonEnum value. Specifies the reason for this event. Its value can be adRsnMoveFirst, adRsnMoveLast, adRsnMoveNext, adRsnMovePrevious, adRsnMove, or adRsnRequery.

pError: An Error object. It describes the error that occurred if the value of *adStatus* is adStatusErrorsOccurred; otherwise it is not set.

adStatus: An EventStatusEnum status value.

pRecordset: The Recordset object for which this event occurred.

RecordChangeComplete

This event is fired after one or more records change.

Syntax

```
RecordChangeComplete adReason, cRecords, pError, adStatus, pRecordset
```

Parameters

adReason: An EventReasonEnum value. Specifies the reason for this event. Its value can be adRsnAddNew, adRsnDelete, adRsnUpdate, adRsnUndoUpdate, adRsnUndoAddNew, adRsnUndoDelete, or adRsnFirstChange.

cRecords: A Long. The number of records changing (affected).

pError: An Error object. It describes the error that occurred if the value of *adStatus* is adStatusErrorsOccurred; otherwise it is not set.

adStatus: An EventStatusEnum status value.

pRecordset: The Recordset object for which this event occurred.

RecordsetChangeComplete

This event is fired after the Recordset has changed.

Syntax

```
RecordsetChangeComplete adReason, pError, adStatus, pRecordset
```

Parameters

adReason: An EventReasonEnum value. Specifies the reason for this event. Its value can be adRsnReQuery, adRsnReSynch, adRsnClose, or adRsnOpen.

adStatus: An EventStatusEnum status value.

pError: An Error object. It describes the error that occurred if the value of *adStatus* in adStatusErrorsOccurred; otherwise it is not set.

pRecordset: The Recordset object for which this event occurred.

WillChangeField

This event is fired before a pending operation changes the value of one or more Field objects in the Recordset.

Syntax

```
WillChangeField cFields, Fields, adStatus, pRecordset
```

Parameters

cFields: A Long. The number of Field objects in *Fields*.

Fields: An array of Variants. Contains Field objects with pending changes.

adStatus: An EventStatusEnum status value.

pRecordset: The Recordset object for which this event occurred.

WillChangeRecord

This event is fired before one or more records (rows) in the Recordset change.

Syntax

```
WillChangeRecord adReason, cRecords, adStatus, pRecordset
```

Parameters

adReason: An EventReasonEnum value. Specifies the reason for this event. Its value can be adRsnAddNew, adRsnDelete, adRsnUpdate, adRsnUndoUpdate, adRsnUndoAddNew, adRsnUndoDelete, or adRsnFirstChange.

cRecords: A Long. The number of records changing (affected).

adStatus: An EventStatusEnum status value.

pRecordset: The Recordset object for which this event occurred.

WillChangeRecordset

This event is fired before a pending operation changes the Recordset.

Syntax

```
WillChangeRecordset adReason, adStatus, pRecordset
```

Parameters

adReason: An `EventReasonEnum` value. Specifies the reason for this event. Its value can be `adRsnReQuery`, `adRsnReSync`, `adRsnClose`, `adRsnOpen`.

adStatus: An `EventStatusEnum` status value.

pRecordset: The Recordset object for which this event occurred.

WillMove

This event is fired before a pending operation changes the current position in the Recordset.

Syntax

```
WillMove adReason, adStatus, pRecordset
```

Parameters

adReason: An `EventReasonEnum` value. Specifies the reason for this event. Its value can be `adRsnMoveFirst`, `adRsnMoveLast`, `adRsnMoveNext`, `adRsnMovePrevious`, `adRsnMove`, or `adRsnRequery`.

adStatus: An `EventStatusEnum` status value.

pRecordset: The Recordset object for which this event occurred.

Stream Object

A Stream object represents a stream of binary data or text.

Methods

Cancel

Cancels execution of a pending, asynchronous method call.

Syntax

```
stream.Cancel
```

Close

Closes an open object and any dependent objects.

Syntax

```
stream.Close
```

CopyTo

Copies the specified number of characters or bytes (depending on Type) in the Stream to another Stream object.

Syntax

```
stream.CopyTo DestStream, NumChars
```

Parameters

DestStream: An object variable value that contains a reference to an open Stream object. The current Stream is copied to the destination Stream specified by *DestStream*. The destination Stream must already be open. If not, a run-time error occurs. Note that the *DestStream* parameter may not be a proxy of Stream object because this requires access to a private interface on the Stream object that cannot be remoted to the client.

NumChars (optional): An Integer value that specifies the number of bytes or characters to be copied from the current position in the source Stream to the destination Stream. The default value is –1, which specifies that all characters or bytes are copied from the current position to EOS.

Flush

Forces the contents of the Stream remaining in the ADO buffer to the underlying object with which the Stream is associated.

Syntax

```
stream.Flush
```

LoadFromFile

Loads the contents of an existing file into a Stream.

Syntax

```
stream.LoadFromFile FileName
```

Parameter

FileName: A String value that contains the name of a file to be loaded into the Stream. *FileName* can contain any valid path and name in UNC format. If the specified file does not exist, a run-time error occurs.

Open

Opens a Stream object to manipulate streams of binary or text data.

Syntax

```
stream.Open Source, Mode, OpenOptions, UserName, Password
```

Parameters

Source (optional): A Variant value that specifies the source of data for the Stream. *Source* may contain an absolute URL string that points to an existing node in a well-known tree structure, such as an e-mail or file system. A URL should be specified using the URL keyword (`URL=http://server/folder`). Alternately, *Source* may contain a reference to an already open Record object, which opens the default stream associated with the Record. If *Source* is not specified, a Stream is instantiated and opened, associated with no underlying source by default.

Mode (optional): A ConnectModeEnum value that specifies the access mode for the resultant Stream (for example, read/write or read-only). Default value is adModeUnknown. See the information on the Mode property, which is coming up, for more information about access modes. If *Mode* is not specified, it is inherited by the source object. For example, if the source Record is opened in read-only mode, the Stream will also be opened in read-only mode by default.

OpenOptions (optional): A StreamOpenOptionsEnum value. Default value is adOpenStreamUnspecified.

UserName (optional): A String value that contains the user identification that, if needed, accesses the Stream object.

Password (optional): A String value that contains the password that, if needed, accesses the Stream object.

Read

Reads a specified number of bytes from a binary Stream object.

Syntax

```
Variant = Stream.Read ( NumBytes )
```

Parameter

NumBytes (optional): A Long value that specifies the number of bytes to read from the file or the StreamReadEnum value adReadAll, which is the default.

Return Value

The Read method reads a specified number of bytes or the entire stream from a Stream object and returns the resulting data as a Variant.

ReadText

Reads specified number of characters from a text Stream object.

Syntax

```
String = stream.ReadText ( NumChars )
```

Parameter

NumChars (optional): A Long value that specifies the number of characters to read from the file, or a StreamReadEnum value. The default value is adReadAll.

Return Value

The ReadText method reads a specified number of characters, an entire line, or the entire stream from a Stream object and returns the resulting string.

SaveToFile

Saves the binary contents of a Stream to a file.

Syntax

```
stream.SaveToFile FileName, SaveOptions
```

Parameters

FileName: A String value that contains the fully-qualified name of the file to which the contents of the Stream will be saved. You can save to any valid local location, or any location you have access to via a UNC value.

SaveOptions: A SaveOptionsEnum value that specifies whether a new file should be created by SaveToFile, if it does not already exist. Default value is adSaveCreateNotExists. With these options you can specify that an error occurs if the specified file does not exist. You can also specify that SaveToFile overwrites the current contents of an existing file.

SetEOS

Sets the position that is the end of the stream.

Syntax

```
stream.SetEOS
```

SkipLine

Skips one entire line when reading a text stream.

Syntax

```
stream.SkipLine
```

Write

Writes binary data to a Stream object.

Syntax

```
stream.Write Buffer
```

Parameter

Buffer: A Variant that contains an array of bytes to be written.

WriteText

Writes a specified text string to a Stream object.

Syntax

```
stream.WriteText Data, Options
```

Parameters

Data: A String value that contains the text in characters to be written.

Options (optional): A StreamWriteEnum value that specifies whether a line separator character must be written at the end of the specified string.

Properties

Charset

Indicates the character set into which the contents of a text Stream should be translated.

Settings and Return Values

Sets or returns a String value that specifies the character set into which the contents of the Stream will be translated. The default value is `Unicode`. Allowed values are typical strings passed over the interface as Internet character set strings (for example, `iso-8859-1`, `Windows-1252`, etc.). For a list of the character set strings that is known by a system, see the subkeys of HKEY_CLASSES_ROOT\MIME\Database\Charset in the Windows Registry.

EOS

Indicates whether the current position is at the end of the stream.

Return Values

Returns a Boolean value that indicates whether the current position is at the end of the stream. EOS returns True if there are no more bytes in the stream; it returns False if there are more bytes following the current position.

LineSeparator

Indicates the binary character to be used as the line separator in text Stream objects.

Settings and Return Values

Sets or returns a `LineSeparatorEnum` value that indicates the line separator character used in the Stream. The default value is `adCRLF`.

Mode

Indicates the available permissions for modifying data in a Stream object.

Settings and Return Values

Sets or returns a `ConnectModeEnum` value. The default value for a Stream associated with an underlying source (opened with a URL as the source, or as the default Stream of a Record) is `adReadOnly`. The default value for a Stream not associated with an underlying source (instantiated in memory) is `adModeUnknown`.

Position

Indicates the current position within a Stream object.

Settings and Return Values

Sets or returns a Long value that specifies the offset, in number of bytes, of the current position from the beginning of the stream. The default is 0, which represents the first byte in the stream.

Size

Indicates the size of the stream in number of bytes.

Return Values

Returns a Long value that specifies the size of the stream in number of bytes. The default value is the size of the stream, or -1 if the size of the stream is not known.

State

Indicates for all applicable objects whether the state of the object is open or closed.

Indicates for all applicable objects executing an asynchronous method, whether the current state of the object is connecting, executing, or retrieving.

Return Value

Returns a Long value that can be an `ObjectStateEnum` value. The default value is `adStateClosed`.

Type

Indicates the type of data contained in the Stream (binary or text).

Settings and Return Values

Sets or returns a `StreamTypeEnum` value that specifies the type of data contained in the Stream object. The default value is `adTypeText`. However, if binary data is initially written to a new, empty Stream, the Type will be changed to `adTypeBinary`.

Enumerated Constants

ADCPROP_ASYNCTHREADPRIORITY_ENUM

For an RDS Recordset object, this constant specifies the execution priority of the asynchronous thread that retrieves data. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adPriorityAboveNormal	4	Sets priority between normal and highest.
adPriorityBelowNormal	2	Sets priority between lowest and normal.
adPriorityHighest	5	Sets priority to the highest possible.
adPriorityLowest	1	Sets priority to the lowest possible.
adPriorityNormal	3	Sets priority to normal.

ADCPROP_AUTORECALC_ENUM

This constant specifies when the MSDataShape provider re-calculates aggregate and calculated columns in a hierarchical Recordset.

These constants are only used with the MSDataShape provider and the Recordset `Auto Recalc` dynamic property. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adRecalcAlways	1	Default. Recalculates whenever the MSDataShape provider determines values that the calculated columns depend upon have changed.
adRecalcUpFront	0	Calculates only when initially building the hierarchical Recordset.

ADCPROP_UPDATECRITERIA_ENUM

This constant specifies which fields can be used to detect conflicts during an optimistic update of a row of the data source with a Recordset object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCriteriaAllCols	1	Detects conflicts if any column of the data source row has been changed.
adCriteriaKey	0	Detects conflicts if the key column of the data source row has been changed, which means that the row has been deleted.
adCriteriaTimeStamp	3	Detects conflicts if the timestamp of the data source row has been changed, which means the row has been accessed after the Recordset was obtained.
adCriteriaUpdCols	2	Detects conflicts if any of the columns of the data source row that correspond to updated fields of the Recordset have been changed.

ADCPROP_UPDATERESYNC_ENUM

This constant specifies whether the `UpdateBatch` method is followed by an implicit `Resync` method operation and if so, the scope of that operation. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adResyncAll	15	Invokes `Resync` for each row with pending changes.
adResyncAutoIncrement	1	Default. Attempts to retrieve the new identity value for columns that are automatically incremented or generated by the data source, such as Microsoft Jet AutoNumber fields or Microsoft SQL Server Identity columns.
adResyncConflicts	2	Invokes `Resync` for all rows in which the update or delete operation failed because of a concurrency conflict.
adResyncInserts	8	Invokes `Resync` for all successfully inserted rows. However, primary key column values are not resynchronized. Instead, contents of newly inserted rows are resynchronized based on the existing primary key value. If the primary key value has changed, `Resync` won't retrieve the contents of the intended row. For automatically incrementing primary key values, first call `UpdateBatch` with `adResyncAutoIncrement` to retrieve the data source-generated primary key value.

(continued)

CONSTANT	VALUE	DESCRIPTION
adResyncNone	0	Does not invoke `Resync`.
adResyncUpdates	4	Invokes `Resync` for all successfully updated rows.

AffectEnum

This constant specifies which records are affected by an operation. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adAffectAll	3	If there is not a Filter applied to the Recordset, affects all records. If the Filter property is set to a string criteria (such as `"Author='Smith"`, then the operation affects visible records in the current chapter. If the Filter property is set to a member of the FilterGroupEnum or an array of Bookmarks, then the operation will affect all rows of the Recordset. Note: `adAffectAll` is hidden in the Visual Basic Object Browser.
adAffectAllChapters	4	Affects all records in all sibling chapters of the Recordset, including those not visible via any Filter that is currently applied.
adAffectCurrent	1	Affects only the current record.
adAffectGroup	2	Affects only records that satisfy the current Filter property setting. You must set the `Filter` property to a `FilterGroupEnum` value or an array of Bookmarks to use this option.

BookmarkEnum

This constant specifies a bookmark indicating where the operation should begin. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adBookmarkCurrent	0	Starts at the current record.
adBookmarkFirst	1	Starts at the first record.
adBookmarkLast	2	Starts at the last record.

CommandTypeEnum

This constant specifies how a command argument should be interpreted. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCmdUnspecified	–1	Does not specify the command type argument.
adCmdText	1	Evaluates `CommandText` as a textual definition of a command or stored procedure call.
adCmdTable	2	Evaluates `CommandText` as a table name whose columns are all returned by an internally generated SQL query.
adCmdStoredProc	4	Evaluates `CommandText` as a stored procedure name.
adCmdUnknown	8	Default. Indicates that the type of command in the `CommandText` property is not known.
adCmdFile	256	Evaluates `CommandText` as the file name of a persistently stored Recordset.
adCmdTableDirect	512	Evaluates `CommandText` as a table name whose columns are all returned.

CompareEnum

This constant specifies the relative position of two records represented by their bookmarks. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCompareEqual	1	Indicates that the bookmarks are equal.
adCompareGreaterThan	2	Indicates that the first bookmark is after the second.
adCompareLessThan	0	Indicates that the first bookmark is before the second.
adCompareNotComparable	4	Indicates that the bookmarks cannot be compared.
adCompareNotEqual	3	Indicates that the bookmarks are not equal and not ordered.

ConnectModeEnum

This constant specifies the available permissions for modifying data in a Connection, opening a Record, or specifying values for the `Mode` property of the Record and Stream objects. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adModeRead	1	Indicates read-only permissions.
adModeReadWrite	3	Indicates read/write permissions.
adModeRecursive	0x400000	Used in conjunction with the other **ShareDeny** values (`adModeShareDenyNone`, `adModeShareDenyWrite`, or `adModeShareDenyRead`) to propagate sharing restrictions to all sub-records of the current Record. It has no affect if the Record does not have any children. A run-time error is generated if it is used with `adModeShareDenyNone` only. However, it can be used with `adModeShareDenyNone` when combined with other values. For example, you can use `"adModeRead Or adModeShareDenyNone Or adModeRecursive"`.
adModeShareDenyNone	16	Allows others to open a connection with any permissions. Neither read nor write access can be denied to others.
adModeShareDenyRead	4	Prevents others from opening a connection with read permissions.
adModeShareDenyWrite	8	Prevents others from opening a connection with write permissions.
adModeShareExclusive	12	Prevents others from opening a connection.
adModeUnknown	0	Default. Indicates that the permissions have not yet been set or cannot be determined.
adModeWrite	2	Indicates write-only permissions.

ConnectOptionEnum

This constant specifies whether the `Open` method of a Connection object should return after (synchronously) or before (asynchronously) the connection is established. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adAsyncConnect	16	Opens the connection asynchronously. The `ConnectComplete` event may be used to determine when the connection is available.
adConnectUnspecified	–1	Default. Opens the connection synchronously.

ConnectPromptEnum

This constant specifies whether a dialog box should be displayed to prompt for missing parameters when opening a connection to a data source. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adPromptAlways	1	Prompts always.
adPromptComplete	2	Prompts if more information is required.
adPromptCompleteRequired	3	Prompts if more information is required but optional parameters are not allowed.
adPromptNever	4	Never prompts.

CopyRecordOptionsEnum

This constant specifies the behavior of the `CopyRecord` method. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCopyAllowEmulation	4	Indicates that the *`Source`* provider attempts to simulate the copy using download and upload operations if this method fails due to *`Destination`* being on a different server or is serviced by a provider other than *`Source`*. Note that differing provider capabilities may hamper performance or cause a loss of data.
adCopyNonRecursive	2	Copies the current directory, but none of its subdirectories, to the destination. The copy operation is not recursive.
adCopyOverWrite	1	Overwrites the file or directory if the *`Destination`* points to an existing file or directory.

(continued)

CONSTANT	VALUE	DESCRIPTION
adCopyUnspecified	-1	Default. Performs the default copy operation. The operation fails if the destination file or directory already exists, and the operation copies recursively.

CursorLocationEnum

This constant specifies the location of the cursor service. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adUseClient	3	Uses client-side cursors supplied by a local cursor library. Local cursor services often allow many features that driver-supplied cursors will not, so using this setting may provide an advantage with respect to features that will be enabled. For backward compatibility, the synonym `adUseClientBatch` is also supported.
adUseNone	1	Does not use cursor services. (This constant is obsolete and appears solely for the sake of backward compatibility.)
adUseServer	2	Default. Uses data-provider or driver-supplied cursors. These cursors are sometimes very flexible and allow for additional sensitivity to changes others make to the data source. However, some features of the Microsoft Cursor Service for OLE DB (such as disassociated Recordset objects) cannot be simulated with server-side cursors and these features will be unavailable with this setting.

CursorOptionEnum

This constant specifies which functionality the `Supports` method should test for. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adAddNew	0x1000400	Supports the `AddNew` method to add new records.
adApproxPosition	0x4000	Supports the `AbsolutePosition` and `AbsolutePage` properties.
adBookmark	0x2000	Supports the `Bookmark` property to gain access to specific records.
adDelete	0x1000800	Supports the `Delete` method to delete records.
adFind	0x80000	Supports the `Find` method to locate a row in a Recordset.
adHoldRecords	0x100	Retrieves more records or changes the next position without committing all pending changes.
adIndex	0x100000	Supports the `Index` property to name an index.
adMovePrevious	0x200	Supports the `MoveFirst` and `MovePrevious` methods, and `Move` or `GetRows` methods to move the current record position backward without requiring bookmarks.
adNotify	0x40000	Indicates that the underlying data provider supports notifications (which determines whether Recordset events are supported).
adResync	0x20000	Supports the `Resync` method to update the cursor with the data that is visible in the underlying database.
adSeek	0x200000	Supports the `Seek` method to locate a row in a Recordset.
adUpdate	0x1008000	Supports the `Update` method to modify existing data.
adUpdateBatch	0x10000	Supports batch updating (`UpdateBatch` and `CancelBatch` methods) to transmit groups of changes to the provider.

CursorTypeEnum

This constant specifies the type of cursor used in a `Recordset` object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adOpenDynamic	2	Uses a dynamic cursor. Additions, changes, and deletions by other users are visible, and all types of movement through the Recordset are allowed, except for bookmarks, if the provider doesn't support them.
adOpenForwardOnly	0	Default. Uses a forward-only cursor. Identical to a static cursor, except that you can only scroll forward through records. This improves performance when you need to make only one pass through a Recordset.
adOpenKeyset	1	Uses a keyset cursor. Like a dynamic cursor, except that you can't see records that other users add, although records that other users delete are inaccessible from your Recordset. Data changes by other users are still visible.
adOpenStatic	3	Uses a static cursor. A static copy of a set of records that you can use to find data or generate reports. Additions, changes, or deletions by other users are not visible.
adOpenUnspecified	-1	Does not specify the type of cursor.

DataTypeEnum

This specifies the data type of a Field, Parameter, or Property. The constants are shown in the following table.

The corresponding OLE DB type indicator is shown in parentheses in the Description column of the table. For more information about OLE DB data types, see Chapter 13 of this book and Appendix A of the *OLE DB Programmer's Reference*.

CONSTANT	VALUE	DESCRIPTION
adArray (does not apply to ADOX)	0x2000	A flag value, always combined with another data type constant, that indicates an array of that other data type.
adBigInt	20	Indicates an eight-byte signed integer (DBTYPE_I8).
adBinary	128	Indicates a binary value (DBTYPE_BYTES).
adBoolean	11	Indicates a boolean value (DBTYPE_BOOL).
adBSTR	8	Indicates a null-terminated character string (Unicode) (DBTYPE_BSTR).
adChapter	136	Indicates a four-byte chapter value that identifies rows in a child rowset (DBTYPE_HCHAPTER).
adChar	129	Indicates a string value (DBTYPE_STR).
adCurrency	6	Indicates a currency value (DBTYPE_CY). Currency is a fixed-point number with four digits to the right of the decimal point. It is stored in an eight-byte signed integer scaled by 10,000.
adDate	7	Indicates a date value (DBTYPE_DATE). A date is stored as a double, the whole part of which is the number of days since December 30, 1899, and the fractional part of which is the fraction of a day.
adDBDate	133	Indicates a date value (yyyymmdd) (DBTYPE_DBDATE).
adDBTime	134	Indicates a time value (hhmmss) (DBTYPE_DBTIME).
adDBTimeStamp	135	Indicates a date/time stamp (yyyymmddhhmmss plus a fraction in billionths) (DBTYPE_DBTIMESTAMP).
adDecimal	14	Indicates an exact numeric value with a fixed precision and scale (DBTYPE_DECIMAL).
adDouble	5	Indicates a double-precision floating-point value (DBTYPE_R8).
adEmpty	0	Specifies no value (DBTYPE_EMPTY).

(continued)

CONSTANT	VALUE	DESCRIPTION
adError	10	Indicates a 32-bit error code (DBTYPE_ERROR).
adFileTime	64	Indicates a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (DBTYPE_FILETIME).
adGUID	72	Indicates a globally unique identifier (GUID) (DBTYPE_GUID).
adIDispatch	9	Indicates a pointer to an IDispatch interface on a COM object (DBTYPE_IDISPATCH). Note: This data type is currently not supported by ADO. Usage may cause unpredictable results.
adInteger	3	Indicates a four-byte signed integer (DBTYPE_I4).
adIUnknown	13	Indicates a pointer to an IUnknown interface on a COM object (DBTYPE_IUNKNOWN). Note: This data type is currently not supported by ADO. Usage may cause unpredictable results.
adLongVarBinary	205	Indicates a long binary value (Parameter object only).
adLongVarChar	201	Indicates a long string value (Parameter object only).
adLongVarWChar	203	Indicates a long null-terminated Unicode string value (Parameter object only).
adNumeric	131	Indicates an exact numeric value with a fixed precision and scale (DBTYPE_NUMERIC).
adPropVariant	138	Indicates an Automation PROPVARIANT (DBTYPE_PROP_VARIANT).
adSingle	4	Indicates a single-precision floating-point value (DBTYPE_R4).
adSmallInt	2	Indicates a two-byte signed integer (DBTYPE_I2).
adTinyInt	16	Indicates a one-byte signed integer (DBTYPE_I1).
adUnsignedBigInt	21	Indicates an eight-byte unsigned integer (DBTYPE_UI8).
adUnsignedInt	19	Indicates a four-byte unsigned integer (DBTYPE_UI4).

(continued)

CONSTANT	VALUE	DESCRIPTION
adUnsignedSmallInt	18	Indicates a two-byte unsigned integer (DBTYPE_UI2).
adUnsignedTinyInt	17	Indicates a one-byte unsigned integer (DBTYPE_UI1).
adUserDefined	132	Indicates a user-defined variable (DBTYPE_UDT).
adVarBinary	204	Indicates a binary value (Parameter object only).
adVarChar	200	Indicates a string value (Parameter object only).
adVariant	12	Indicates an Automation Variant (DBTYPE_VARIANT). Note: This data type is currently not supported by ADO. Usage may cause unpredictable results.
adVarNumeric	139	Indicates a numeric value (Parameter object only).
adVarWChar	202	Indicates a null-terminated Unicode character string (Parameter object only).
adWChar	130	Indicates a null-terminated Unicode character string (DBTYPE_WSTR).

EditModeEnum

This constant specifies the editing status of a record. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adEditNone	0	Indicates that no editing operation is in progress.
adEditInProgress	1	Indicates that data in the current record has been modified but not saved.
adEditAdd	2	Indicates that the `AddNew` method has been called, and the current record in the copy buffer is a new record that has not been saved in the database.
adEditDelete	4	Indicates that the current record has been deleted.

ErrorValueEnum

This constant specifies the type of ADO run-time error.

Three forms of the error number are listed in the table that appears in this section:

- Positive decimal: The low two bytes of the full number in decimal format. This number is displayed in the default Visual Basic error message dialog box. For example: Run-time error '3707'.
- Negative decimal: The decimal translation of the full error number.
- Hexadecimal: The hexadecimal representation of the full error number. The Windows facility code is in the fourth digit. The facility code for ADO error numbers is *A*. For example: 0x800*A*0E7B.

CONSTANT	VALUE	DESCRIPTION
adErrBoundToCommand	3707 -2146824581 0x800A0E7B	Cannot change the ActiveConnection property of a Recordset object which has a Command object as its source.
adErrCannotComplete	3732 -2146824556 0x800A0E94	Server cannot complete the operation.
adErrCantChangeConnection	3748 -2146824540 0x800A0EA4	Connection was denied. New connection you requested has different characteristics than the one already in use.
adErrCantChangeProvider	3220 -2146825068 0x800A0C94	Supplied provider is different from the one already in use.
adErrCantConvertvalue	3724 -2146824564 0x800A0E8C	Data value cannot be converted for reasons other than sign mismatch or data overflow. For example, conversion would have truncated data.
adErrCantCreate	3725 -2146824563 0x800A0E8D	Data value cannot be set or retrieved because the field data type was unknown, or the provider had insufficient resources to perform the operation.

(continued)

CONSTANT	VALUE	DESCRIPTION
adErrCatalogNotSet	3747 -2146824541 0x800A0EA3	Operation requires a valid ParentCatalog.
adErrColumnNotOnThisRow	3726 -2146824562 0x800A0E8E	Record does not contain this field.
adErrDataConversion	3421 -2146824867 0x800A0D5D	Application uses a value of the wrong type for the current operation.
adErrDataOverflow	3721 -2146824567 0x800A0E89	Data value is too large to be represented by the field data type.
adErrDelResOutOfScope	3738 -2146824550 0x800A0E9A	URL of the object to be deleted is outside the scope of the current record.
adErrDenyNotSupported	3750 -2146824538 0x800A0EA6	Provider does not support sharing restrictions.
adErrDenyTypeNotSupported	3751 -2146824537 0x800A0EA7	Provider does not support the requested kind of sharing restriction.
adErrFeatureNotAvailable	3251 -2146825037 0x800A0CB3	Object or provider is not capable of performing requested operation.
adErrFieldsUpdateFailed	3749 -2146824539 0x800A0EA5	Fields update failed. For further information, examine the `Status` property of individual field objects.
adErrIllegalOperation	3219 -2146825069 0x800A0C93	Operation is not allowed in this context.
adErrIntegrityViolation	3719 -2146824569 0x800A0E87	Data value conflicts with the integrity constraints of the field.
adErrInTransaction	3246 -2146825042 0x800A0CAE	Connection object cannot be explicitly closed while in a transaction.

(continued)

CONSTANT	VALUE	DESCRIPTION
adErrInvalidArgument	3001 -2146825287 0x800A0BB9	Arguments are of the wrong type, are out of acceptable range, or are in conflict with one another.
adErrInvalidConnection	3709 -2146824579 0x800A0E7D	Operation is not allowed on an object referencing a closed or invalid connection.
adErrInvalidParamInfo	3708 -2146824580 0x800A0E7C	Parameter object is improperly defined. Inconsistent or incomplete information was provided.
adErrInvalidTransaction	3714 -2146824574 0x800A0E82	Coordinating transaction is invalid or has not started.
adErrInvalidURL	3729 -2146824559 0x800A0E91	URL contains invalid characters. Make sure the URL is typed correctly.
adErrItemNotFound	3265 -2146825023 0x800A0CC1	Item cannot be found in the collection corresponding to the requested name or ordinal.
adErrNoCurrentRecord	3021 -2146825267 0x800A0BCD	Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record.
adErrNotExecuting	3715 -2146824573 0x800A0E83	Operation cannot be performed while not executing.
adErrNotReentrant	3710 -2146824578 0x800A0E7E	Operation cannot be performed while processing event.
adErrObjectClosed	3704 -2146824584 0x800A0E78	Operation is not allowed when the object is closed.
adErrObjectInCollection	3367 -2146824921 0x800A0D27	Object is already in collection. Cannot append.
adErrObjectNotSet	3420 -2146824868 0x800A0D5C	Object is no longer valid.

(continued)

CONSTANT	VALUE	DESCRIPTION
adErrObjectOpen	3705 -2146824583 0x800A0E79	Operation is not allowed when the object is open.
adErrOpeningFile	3002 -2146825286 0x800A0BBA	File could not be opened.
adErrOperationCancelled	3712 -2146824576 0x800A0E80	Operation has been cancelled by the user.
adErrOutOfSpace	3734 -2146824554 0x800A0E96	Operation cannot be performed. Provider cannot obtain enough storage space.
adErrPermissionDenied	3720 -2146824568 0x800A0E88	Insufficient permission prevents writing to the field.
adErrPropConflicting	3742 -2146824546 0x800A0E9E	Property value conflicts with a related property.
adErrPropInvalidColumn	3739 -2146824549 0x800A0E9B	Property cannot apply to the specified field.
adErrPropInvalidOption	3740 -2146824548 0x800A0E9C	Property attribute is invalid.
adErrPropInvalidValue	3741 -2146824547 0x800A0E9D	Property value is invalid. Make sure the value is typed correctly.
adErrPropNotAllSettable	3743 -2146824545 0x800A0E9F	Property is read-only or cannot be set.
adErrPropNotSet	3744 -2146824544 0x800A0EA0	Optional property value was not set.
adErrPropNotSettable	3745 -2146824543 0x800A0EA1	Read-only property value was not set.

(continued)

CONSTANT	VALUE	DESCRIPTION
adErrPropNotSupported	3746 -2146824542 0x800A0EA2	Provider does not support the property.
adErrProviderFailed	3000 -2146825288 0x800A0BB8	Provider failed to perform the requested operation.
adErrProviderNotFound	3706 -2146824582 0x800A0E7A	Provider cannot be found. It may not be properly installed.
adErrReadFile	3003 -2146825285 0x800A0BBB	File could not be read.
adErrResourceExists	3731 -2146824557 0x800A0E93	Copy operation cannot be performed. Object named by destination URL already exists. Specify `adCopyOverwrite` to replace the object.
adErrResourceLocked	3730 -2146824558 0x800A0E92	Object represented by the specified URL is locked by one or more other processes. Wait until the process has finished and attempt the operation again.
adErrResourceOutOfScope	3735 -2146824553 0x800A0E97	Source or destination URL is outside the scope of the current record.
adErrSchemaViolation	3722 -2146824566 0x800A0E8A	Data value conflicts with the data type or constraints of the field.
adErrSignMismatch	3723 -2146824565 0x800A0E8B	Conversion failed because the data value was signed and the field data type used by the provider was unsigned.
adErrStillConnecting	3713 -2146824575 0x800A0E81	Operation cannot be performed while connecting aynchronously.

(continued)

CONSTANT	VALUE	DESCRIPTION
adErrStillExecuting	3711 -2146824577 0x800A0E7F	Operation cannot be performed while executing asynchronously.
adErrTreePermissionDenied	3728 -2146824560 0x800A0E90	Permissions are insufficient to access tree or subtree.
adErrUnavailable	3736 -2146824552 0x800A0E98	Operation failed to complete and the status is unavailable. The field may be unavailable or the operation was not attempted.
adErrUnsafeOperation	3716 -2146824572 0x800A0E84	Safety settings on this computer prohibit accessing a data source on another domain.
adErrURLDoesNotExist	3727 -2146824561 0x800A0E8F	Either the source URL or the parent of the destination URL does not exist.
adErrURLNamedRow DoesNotExist	3737 -2146824551 0x800A0E99	Record named by this URL does not exist.
adErrVolumeNotFound	3733 -2146824555 0x800A0E95	Provider cannot locate the storage device indicated by the URL. Make sure the URL is typed correctly.
adErrWriteFile	3004 -2146825284 0x800A0BBC	Write to file failed.
adWrnSecurityDialog	3717 -2146824571 0x800A0E85	For internal use only. Do not use.
adWrnSecurityDialogHeader	3718 -2146824570 0x800A0E86	For internal use only. Do not use.

EventReasonEnum

This constant specifies the reason that caused an event to occur. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adRsnAddNew	1	An operation added a new record.
adRsnClose	9	An operation closed the Recordset.
adRsnDelete	2	An operation deleted a record.
adRsnFirstChange	11	An operation made the first change to a record.
adRsnMove	10	An operation moved the record pointer within the Recordset.
adRsnMoveFirst	12	An operation moved the record pointer to the first record in the Recordset.
adRsnMoveLast	15	An operation moved the record pointer to the last record in the Recordset.
adRsnMoveNext	13	An operation moved the record pointer to the next record in the Recordset.
adRsnMovePrevious	14	An operation moved the record pointer to the previous record in the Recordset.
adRsnRequery	7	An operation requeried the Recordset.
adRsnResynch	8	An operation resynchronized the Recordset with the database.
adRsnUndoAddNew	5	An operation reversed the addition of a new record.
adRsnUndoDelete	6	An operation reversed the deletion of a record.
adRsnUndoUpdate	4	An operation reversed the update of a record.
adRsnUpdate	3	An operation updated an existing record.

EventStatusEnum

This constant specifies the current status of the execution of an event. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adStatusCancel	4	Requests cancellation of the operation that caused the event to occur.
adStatusCantDeny	3	Indicates that the operation cannot request cancellation of the pending operation.
adStatusErrorsOccurred	2	Indicates that the operation that caused the event failed due to an error or errors.
adStatusOK	1	Indicates that the operation that caused the event was successful.
adStatusUnwantedEvent	5	Prevents subsequent notifications before the event method has finished executing.

ExecuteOptionEnum

This constant specifies how a provider should execute a command. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adAsyncExecute	0x10	Indicates that the command should execute asynchronously.
adAsyncFetch	0x20	Indicates that the remaining rows after the initial quantity specified in the `CacheSize` property should be retrieved asynchronously.
adAsyncFetchNonBlocking	0x40	Indicates that the main thread never blocks while retrieving. If the requested row has not been retrieved, the current row automatically moves to the end of the file. If you open a Recordset from a Stream containing a persistently stored Recordset, `adAsyncFetchNonBlocking` will have no effect; the operation will be synchronous and blocking. `adAsynchFetchNonBlocking` has no effect when the `adCmdTableDirect` option is used to open the Recordset.

(continued)

CONSTANT	VALUE	DESCRIPTION
adExecuteNoRecords	0x80	Indicates that the command text is a command or stored procedure that does not return rows (for example, a command that only inserts data). If any rows are retrieved, they are discarded and not returned. Always combined with `CommandTypeEnum` values `adCmdText` or `adCmdStoredProc`. `adExecuteNoRecords` can only be passed as an optional parameter to the `Command` or `Connection Execute` method. Using it as an argument to the Command object `CommandType` property will generate an error.
adOptionUnspecified	–1	Indicates that the command is unspecified.

FieldEnum

This constant specifies the special fields referenced in a Record object's Fields collection. The constants are shown in the following table.

These constants provide a "shortcut" to accessing special fields associated with a Record. Retrieve the Field object from the Fields collection, and then obtain its contents with the Field object's `Value` property.

CONSTANT	VALUE	DESCRIPTION
adDefaultStream	–1	References the field containing the default Stream object associated with a Record.
adRecordURL	–2	References the field containing the absolute URL string for the current Record.

FieldAttributeEnum

This constant specifies one or more attributes of a Field object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adFldCacheDeferred	0x1000	Indicates that the provider caches field values and that subsequent reads are done from the cache.

(continued)

CONSTANT	VALUE	DESCRIPTION
adFldFixed	0x10	Indicates that the field contains fixed-length data.
adFldIsChapter	0x2000	Indicates that the field contains a chapter value, which specifies a specific child recordset related to this parent field. Typically, chapter fields are used with data shaping or filters.
adFldIsCollection	0x40000	Indicates that the field specifies that the resource represented by the record is a collection of other resources, such as a folder, rather than a simple resource, such as a text file.
adFldIsDefaultStream	0x20000	Indicates that the field contains the default stream for the resource represented by the record. For example, the default stream can be the HTML content of a root folder on a Web site, which is automatically served when the root URL is specified.
adFldIsNullable	0x20	Indicates that the field accepts null values.
adFldIsRowURL	0x10000	Indicates that the field contains the URL that names the resource from the data store represented by the record.
adFldKeyColumn	0x8000	Indicates that the field is the primary key of the underlying rowset. Also can indicate that the field is part of a compound primary key.
adFldLong	0x80	Indicates that the field is a long binary field. Also indicates that you can use the `AppendChunk` and `GetChunk` methods.
adFldMayBeNull	0x40	Indicates that you can read null values from the field.
adFldMayDefer	0x2	Indicates that the field is deferred—that is, the field values are not retrieved from the data source with the whole record, but only when you explicitly access them.
adFldNegativeScale	0x4000	Indicates that the field represents a numeric value from a column that supports negative scale values. The scale is specified by the NumericScale property.

(continued)

CONSTANT	VALUE	DESCRIPTION
adFldRowID	0x100	Indicates that the field contains a persistent row identifier that cannot be written to and has no meaningful value except to identify the row (such as a record number, unique identifier, and so forth).
adFldRowVersion	0x200	Indicates that the field contains some kind of time or date stamp used to track updates.
adFldUnknownUpdatable	0x8	Indicates that the provider cannot determine if you can write to the field.
adFldUnspecified	-1	Indicates that the provider does not specify the field attributes.
adFldUpdatable	0x4	Indicates that you can write to the field.

FieldStatusEnum

This constant specifies the status of a Field object. The constants are shown in the following table.

The `adFieldPending` values indicate the operation that caused the status to be set, and may be combined with other status values.

CONSTANT	VALUE	DESCRIPTION
adFieldAlreadyExists	26	Indicates that the specified field already exists.
adFieldBadStatus	12	Indicates that an invalid status value was sent from ADO to the OLE DB provider. Possible causes include an OLE DB 1.0 or 1.1 provider, or an improper combination of Value and Status.
adFieldCannotComplete	20	Indicates that the server of the URL specified by Source could not complete the operation.
adFieldCannotDeleteSource	23	Indicates that during a move operation, a tree or subtree was moved to a new location, but the source could not be deleted.
adFieldCantConvertValue	2	Indicates that the field cannot be retrieved or stored without loss of data.
adFieldCantCreate	7	Indicates that the field could not be added because the provider exceeded a limitation (such as the number of fields allowed).

(continued)

CONSTANT	VALUE	DESCRIPTION
adFieldDataOverflow	6	Indicates that the data returned from the provider overflowed the data type of the field.
adFieldDefault	13	Indicates that the default value for the field was used when setting data.
adFieldDoesNotExist	16	Indicates that the field specified does not exist.
adFieldIgnore	15	Indicates that this field was skipped when setting data values in the source. No value was set by the provider.
adFieldIntegrityViolation	10	Indicates that the field cannot be modified because it is a calculated or derived entity.
adFieldInvalidURL	17	Indicates that the data source URL contains invalid characters.
adFieldIsNull	3	Indicates that the provider returned a null value.
adFieldOK	0	Default. Indicates that the field was successfully added or deleted.
adFieldOutOfSpace	22	Indicates that the provider is unable to obtain enough storage space to complete a move or copy operation.
adFieldPendingChange	0x40000	Indicates that either the field has been deleted and then re-added, perhaps with a different data type, or that the value of the field which previously had a status of `adFieldOK` has changed. The final form of the field will modify the Fields collection after the `Update` method is called.
adFieldPendingDelete	0x20000	Indicates that the Delete operation caused the status to be set. The field has been marked for deletion from the Fields collection after the `Update` method is called.
adFieldPendingInsert	0x10000	Indicates that the Append operation caused the status to be set. The Field has been marked to be added to the Fields collection after the `Update` method is called.
adFieldPendingUnknown	0x80000	Indicates that the provider cannot determine what operation caused field status to be set.

(continued)

CONSTANT	VALUE	DESCRIPTION
adFieldPending-UnknownDelete	0x100000	Indicates that the provider cannot determine what operation caused field status to be set, and that the field will be deleted from the Fields collection after the `Update` method is called.
adFieldPermissionDenied	9	Indicates that the field cannot be modified because it is defined as read-only.
adFieldReadOnly	24	Indicates that the field in the data source is defined as read-only.
adFieldResourceExists	19	Indicates that the provider was unable to perform the operation because an object already exists at the destination URL and it is not able to overwrite the object.
adFieldResourceLocked	18	Indicates that the provider was unable to perform the operation because the data source is locked by one or more other applications or processes.
adFieldResourceOutOfScope	25	Indicates that a source or destination URL is outside the scope of the current record.
adFieldSchemaViolation	11	Indicates that the value violated the data source schema constraint for the field.
adFieldSignMismatch	5	Indicates that data value returned by the provider was signed but the data type of the ADO field value was unsigned.
adFieldTruncated	4	Indicates that variable-length data was truncated when reading from the data source.
adFieldUnavailable	8	Indicates that the provider could not determine the value when reading from the data source. For example, the row was just created, the default value for the column was not available, and a new value had not yet been specified.
adFieldVolumeNotFound	21	Indicates that the provider is unable to locate the storage volume indicated by the URL.

FilterGroupEnum

This constant specifies the group of records to be filtered from a Recordset. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adFilterAffectedRecords	2	Filters for viewing only records affected by the last `Delete`, `Resync`, `UpdateBatch`, or `CancelBatch` call.
adFilterConflictingRecords	5	Filters for viewing the records that failed the last batch update.
adFilterFetchedRecords	3	Filters for viewing the records in the current cache—that is, the results of the last call to retrieve records from the database.
adFilterNone	0	Removes the current filter and restores all records for viewing.
adFilterPendingRecords	1	Filters for viewing only records that have changed but have not yet been sent to the server. Applicable only for batch update mode.

GetRowsOptionEnum

This constant specifies how many records to retrieve from a Recordset. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adGetRowsRest	–1	Retrieves the rest of the records in the Recordset, from either the current position or a bookmark specified by the `Start` parameter of the `GetRows` method.

IsolationLevelEnum

This specifies the level of transaction isolation for a Connection object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adXactUnspecified	–1	Indicates that the provider is using a different isolation level than specified, but that the level cannot be determined.
adXactChaos	16	Indicates that pending changes from more highly isolated transactions cannot be overwritten.
adXactBrowse	256	Indicates that from one transaction you can view uncommitted changes in other transactions.
adXactReadUncommitted	256	Same as `adXactBrowse`.
adXactCursorStability	4096	Indicates that from one transaction you can view changes in other transactions only after they have been committed.
adXactReadCommitted	4096	Same as `adXactCursorStability`.
adXactRepeatableRead	65536	Indicates that from one transaction you cannot see changes made in other transactions, but that requerying can retrieve new Recordset objects.
adXactIsolated	1048576	Indicates that transactions are conducted in isolation of other transactions.
adXactSerializable	1048576	Same as `adXactIsolated`.

LineSeparatorsEnum

This constant specifies the character used as a line separator in text Stream objects. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCR	13	Indicates carriage return.
adCRLF	–1	Default. Indicates carriage return line feed.
adLF	10	Indicates line feed.

LockTypeEnum

This constant specifies the type of lock placed on records during editing. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adLockBatchOptimistic	4	Indicates optimistic batch updates. Required for batch update mode.
adLockOptimistic	3	Indicates optimistic locking, record by record. The provider uses optimistic-locking locking records only when you call the `Update` method.
adLockPessimistic	2	Indicates pessimistic locking, record by record. The provider does what is necessary to ensure successful editing of the records, usually by locking records at the data source immediately after editing.
adLockReadOnly	1	Indicates read-only records. You cannot alter the data.
adLockUnspecified	–1	Does not specify a type of lock. For clones, the clone is created with the same lock type as the original.

MarshalOptionsEnum

This constant specifies which records should be returned to the server. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adMarshalAll	0	Default. Returns all rows to the server.
adMarshalModifiedOnly	1	Returns only modified rows to the server.

MoveRecordOptionsEnum

This constant specifies the behavior of the Record object MoveRecord method. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adMoveUnspecified	–1	Default. Performs the default move operation: The operation fails if the destination file or directory already exists, and the operation updates hypertext links.
adMoveOverWrite	1	Overwrites the destination file or directory, even if it already exists.
adMoveDontUpdateLinks	2	Modifies the default behavior of `MoveRecord` method by not updating the hypertext links of the source Record. The default behavior depends on the capabilities of the provider. Move operation updates links if the provider is capable. If the provider cannot fix links or if this value is not specified, then the move succeeds even when links have not been fixed.
adMoveAllowEmulation	4	Requests that the provider attempt to simulate the move (using download, upload, and delete operations). If the attempt to move the Record fails because the destination URL is on a different server or serviced by a different provider than the source, this may cause increased latency or data loss, due to different provider capabilities when moving resources between providers.

ObjectStateEnum

This constant specifies whether an object is open or closed, connecting to a data source, executing a command, or retrieving data. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adStateClosed	0	Indicates that the object is closed.
adStateOpen	1	Indicates that the object is open.
adStateConnecting	2	Indicates that the object is connecting.
adStateExecuting	4	Indicates that the object is executing a command.
adStateFetching	8	Indicates that the rows of the object are being retrieved.

ParameterAttributesEnum

This constant specifies the attributes of a Parameter object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adParamSigned	16	Indicates that the parameter accepts signed values.
adParamNullable	64	Indicates that the parameter accepts null values.
adParamLong	128	Indicates that the parameter accepts long binary data.

ParameterDirectionEnum

This constant specifies whether the Parameter represents an input parameter, an output parameter, both an input and an output parameter, or the return value from a stored procedure. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adParamInput	1	Default. Indicates that the parameter represents an input parameter.
adParamInputOutput	3	Indicates that the parameter represents both an input and output parameter.
adParamOutput	2	Indicates that the parameter represents an output parameter.
adParamReturnValue	4	Indicates that the parameter represents a return value.
adParamUnknown	0	Indicates that the parameter direction is unknown.

PersistFormatEnum

This constant specifies the format in which to save a Recordset. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adPersistADTG	0	Indicates Microsoft Advanced Data TableGram (ADTG) format.
adPersistXML	1	Indicates Extensible Markup Language (XML) format.

PositionEnum

This constant specifies the current position of the record pointer within a Recordset. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adPosBOF	–2	Indicates that the current record pointer is at BOF (that is, the `BOF` property is True).
adPosEOF	–3	Indicates that the current record pointer is at EOF (that is, the `EOF` property is True).
adPosUnknown	–1	Indicates that the Recordset is empty, the current position is unknown, or the provider does not support the `AbsolutePage` or `AbsolutePosition` property.

PropertyAttributesEnum

This constant specifies the attributes of a Property object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adPropNotSupported	0	Indicates that the property is not supported by the provider.
adPropRequired	1	Indicates that the user must specify a value for this property before the data source is initialized.
adPropOptional	2	Indicates that the user does not need to specify a value for this property before the data source is initialized.
adPropRead	512	Indicates that the user can read the property.
adPropWrite	1024	Indicates that the user can set the property.

RecordCreateOptionsEnum

This constant specifies whether an existing Record should be opened or a new Record created for the Record object Open method. The values can be combined with an AND operator. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adCreateCollection	0x2000	Creates a new Record at the node specified by Source parameter, instead of opening an existing Record. If the source points to an existing node, then a run-time error occurs, unless `adCreateCollection` is combined with `adOpenIfExists` or `adCreateOverwrite`.
adCreateNonCollection	0	Creates a new Record of type `adSimpleRecord`.
adCreateOverwrite	0x4000000	Modifies the creation flags `adCreateCollection`, `adCreateNonCollection`, and `adCreateStructDoc`. When `OR` is used with this value and one of the creation flag values, if the source URL points to an existing node or Record, then the existing Record is overwritten and a new one is created in its place. This value cannot be used together with `adOpenIfExists`.
adCreateStructDoc	0x80000000	Creates a new Record of type `adStructDoc`, instead of opening an existing Record.
adFailIfNotExists	–1	Default. Results in a run-time error if Source points to a nonexistent node.
adOpenIfExists	0x2000000	Modifies the creation flags `adCreateCollection`, `adCreateNonCollection`, and `adCreateStructDoc`. When `OR` is used with this value and one of the creation flag values, if the source URL points to an existing node or Record object, then the provider must open the existing Record instead of creating a new one. This value cannot be used together with `adCreateOverwrite`.

RecordOpenOptionsEnum

This constant specifies options for opening a Record. These values may be combined by using `OR`. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adDelayFetchFields	0x8000	Indicates to the provider that the fields associated with the Record need not be retrieved initially, but can be retrieved at the first attempt to access the field. The default behavior, indicated by the absence of this flag, is to retrieve all the Record object fields.
adDelayFetchStream	0x4000	Indicates to the provider that the default stream associated with the Record need not be retrieved initially. The default behavior, indicated by the absence of this flag, is to retrieve the default stream associated with the Record object.
adOpenAsync	0x1000	Indicates that the Record object is opened in asynchronous mode.
adOpenRecordUnspecified	–1	Default. Indicates no options are specified.
adOpenSource	0x800000	Indicates that if the source points to a node that contains an executable script (such as an .ASP page), then a Record containing the source is opened rather than the executed contents. This value is valid only with non-collection records.

RecordStatusEnum

This constant specifies the status of a record with regard to batch updates and other bulk operations. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adRecCanceled	0x100	Indicates that the record was not saved because the operation was canceled.
adRecCantRelease	0x400	Indicates that the new record was not saved because the existing record was locked.
adRecConcurrencyViolation	0x800	Indicates that the record was not saved because optimistic concurrency was in use.

(continued)

CONSTANT	VALUE	DESCRIPTION
adRecDBDeleted	0x40000	Indicates that the record has already been deleted from the data source.
adRecDeleted	0x4	Indicates that the record was deleted.
adRecIntegrityViolation	0x1000	Indicates that the record was not saved because the user violated integrity constraints.
adRecInvalid	0x10	Indicates that the record was not saved because its bookmark is invalid.
adRecMaxChangesExceeded	0x2000	Indicates that the record was not saved because there were too many pending changes.
adRecModified	0x2	Indicates that the record was modified.
adRecMultipleChanges	0x40	Indicates that the record was not saved because it would have affected multiple records.
adRecNew	0x1	Indicates that the record is new.
adRecObjectOpen	0x4000	Indicates that the record was not saved because of a conflict with an open storage object.
adRecOK	0	Indicates that the record was successfully updated.
adRecOutOfMemory	0x8000	Indicates that the record was not saved because the computer has run out of memory.
adRecPendingChanges	0x80	Indicates that the record was not saved because it refers to a pending insert.
adRecPermissionDenied	0x10000	Indicates that the record was not saved because the user has insufficient permissions.
adRecSchemaViolation	0x20000	Indicates that the record was not saved because it violates the structure of the underlying database.
adRecUnmodified	0x8	Indicates that the record was not modified.

RecordTypeEnum

This constant specifies the type of Record object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adSimpleRecord	0	Indicates a simple record (does not contain child nodes).
adCollectionRecord	1	Indicates a collection record (contains child nodes).
adStructDoc	2	Indicates a special kind of collection record that represents COM structured documents.

ResyncEnum

This constant specifies whether underlying values are overwritten by a call to Resync. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adResyncAllValues	2	Default. Overwrites data, and pending updates are canceled.
adResyncUnderlyingValues	1	Does not overwrite data, and pending updates are not canceled.

SaveOptionsEnum

This constant specifies whether a file should be created or overwritten when saving from a Stream object. The values can be combined with an `AND` operator. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adSaveCreateNotExist	1	Default. Creates a new file if the file specified by the FileName parameter does not already exist.
adSaveCreateOverwrite	4	Overwrites the file with the data from the currently open Stream object, if the file specified by the Filename parameter already exists.

SchemaEnum

This constant specifies the type of schema Recordset that the `OpenSchema` method retrieves. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaAsserts	0	Returns the assertions defined in the catalog that are owned by a given user.	(ASSERTIONS Rowset) CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
adSchemaCatalogs	1	Returns the physical attributes associated with catalogs accessible from the DBMS.	(CATALOGS Rowset) CATALOG_NAME
adSchemaCharacterSets	2	Returns the character sets defined in the catalog that are accessible to a given user.	(CHARACTER_SETS Rowset) CHARACTER_SET_CATALOG CHARACTER_SET_SCHEMA CHARACTER_SET_NAME
adSchemaCheckConstraints	5	Returns the check constraints defined in the catalog that are owned by a given user.	(CHECK_CONSTRAINTS Rowset) CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
adSchemaCollations	3	Returns the character collations defined in the catalog that are accessible to a given user.	(COLLATIONS Rowset) COLLATION_CATALOG COLLATION_SCHEMA COLLATION_NAME
adSchemaColumnPrivileges	13	Returns the privileges on columns of tables defined in the catalog that are available to, or granted by, a given user.	(COLUMN_PRIVILEGES Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME GRANTOR GRANTEE
adSchemaColumns	4	Returns the columns of tables (including views) defined in the catalog that are accessible to a given user.	(COLUMNS Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME

(continued)

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaColumns-DomainUsage	11	Returns the columns defined in the catalog that are dependent on a domain defined in the catalog and owned by a given user.	(COLUMN_DOMAIN_USAGE Rowset) DOMAIN_CATALOG DOMAIN_SCHEMA DOMAIN_NAME COLUMN_NAME
adSchemaConstraint-ColumnUsage	6	Returns the columns used by referential constraints, unique constraints, check constraints, and assertions, defined in the catalog and owned by a given user.	(CONSTRAINT_COLUMN_ USAGE Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME
adSchemaConstraint-TableUsage	7	Returns the tables that are used by referential constraints, unique constraints, check constraints, and assertions defined in the catalog and owned by a given user.	(CONSTRAINT_TABLE_ USAGE Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME
adSchemaCubes	32	Returns information about the available cubes in a schema (or the catalog, if the provider does not support schemas).	(CUBES Rowset*) CATALOG_NAME SCHEMA_NAME CUBE_NAME
adSchemaDBInfoKeywords	30	Returns a list of provider-specific keywords.	(IDBInfo::GetKeywords *) <None>
adSchemaDBInfoLiterals	31	Returns a list of provider-specific literals used in text commands.	(IfDBInfo::GetLiteralInfo *) <None>
adSchemaDimensions	33	Returns information about the dimensions in a given cube. It has one row for each dimension.	(DIMENSIONS Rowset *) CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_NAME DIMENSION_UNIQUE_NAME
adSchemaForeignKeys	27	Returns the foreign key columns defined in the catalog by a given user.	(FOREIGN_KEYS Rowset) PK_TABLE_CATALOG PK_TABLE_SCHEMA PK_TABLE_NAME FK_TABLE_CATALOG FK_TABLE_SCHEMA FK_TABLE_NAME

(continued)

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaHierarchies	34	Returns information about the hierarchies available in a dimension.	(HIERARCHIES Rowset *) CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_NAME HIERARCHY_UNIQUE_NAME
adSchemaIndexes	12	Returns the indexes defined in the catalog that are owned by a given user.	(INDEXES Rowset) TABLE_CATALOG TABLE_SCHEMA INDEX_NAME TYPE TABLE_NAME
adSchemaKeyColumnUsage	8	Returns the columns defined in the catalog that are constrained as keys by a given user.	(KEY_COLUMN_USAGE Rowset) CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME
adSchemaLevels	35	Returns information about the levels available in a dimension.	(LEVELS Rowset*Rowset) CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_NAME LEVEL_UNIQUE_NAME
adSchemaMeasures	36	Returns information about the available measures.	(MEASURES Rowset *) CATALOG_NAME SCHEMA_NAME CUBE_NAME MEASURE_NAME MEASURE_UNIQUE_NAME

(continued)

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaMembers	38	Returns information about the available members.	(MEMBERS Rowset *) CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_UNIQUE_NAME LEVEL_NUMBER MEMBER_NAME MEMBER_UNIQUE_NAME MEMBER_CAPTION MEMBER_TYPE
Tree operator (For more information, see the OLE DB for OLAP)			
adSchemaPrimaryKeys	28	Returns the primary key columns defined in the catalog by a given user.	(PRIMARY_KEYS Rowset) PK_TABLE_CATALOG PK_TABLE_SCHEMA PK_TABLE_NAME
adSchemaProcedureColumns	29	Returns information about the columns of rowsets returned by procedures.	(PROCEDURE_COLUMNS Rowset) PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME COLUMN_NAME
adSchemaProcedure-Parameters	26	Returns information about the parameters and return codes of procedures.	(PROCEDURE_PARAMETERS Rowset) PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME PARAMETER_NAME
adSchemaProcedures	16	Returns the procedures defined in the catalog that are owned by a given user.	(PROCEDURES Rowset) PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME PROCEDURE_TYPE

(continued)

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaProperties	37	Returns information about the available properties for each level of the dimension.	(PROPERTIES Rowset *) CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_UNIQUE_NAME MEMBER_UNIQUE_NAME PROPERTY_TYPE PROPERTY_NAME
adSchemaProviderSpecific	–1	Used if the provider defines its own nonstandard schema queries.	<Provider specific>
adSchemaProviderTypes	22	Returns the (base) data types supported by the data provider.	(PROVIDER_TYPES Rowset) DATA_TYPE BEST_MATCH
adSchemaReferential-Constraints	9	Returns the referential constraints defined in the catalog that are owned by a given user.	(REFERENTIAL_CONSTRAINTS Rowset) CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
adSchemaSchemata	17	Returns the schemas (database objects) that are owned by a given user.	(SCHEMATA Rowset) CATALOG_NAME SCHEMA_NAME SCHEMA_OWNER
adSchemaSQLLanguages	18	Returns the conformance levels, options, and dialects supported by the SQL-implementation processing data defined in the catalog.	(SQL_LANGUAGES Rowset) <None>
adSchemaStatistics	19	Returns the statistics defined in the catalog that are owned by a given user.	(STATISTICS Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME
adSchemaTableConstraints	10	Returns the table constraints defined in the catalog that are owned by a given user.	(TABLE_CONSTRAINTS Rowset) CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME TABLE_CATALOG TABLE_SCHEMA TABLE_NAME CONSTRAINT_TYPE

(continued)

CONSTANT	VALUE	DESCRIPTION	CONSTRAINT COLUMNS
adSchemaTablePrivileges	14	Returns the privileges on tables defined in the catalog that are available to, or granted by, a given user.	(TABLE_PRIVILEGES Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME GRANTOR GRANTEE
adSchemaTables	20	Returns the tables (including views) defined in the catalog that are accessible to a given user.	(TABLES Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME TABLE_TYPE
adSchemaTranslations	21	Returns the character translations defined in the catalog that are accessible to a given user.	(TRANSLATIONS Rowset) TRANSLATION_CATALOG TRANSLATION_SCHEMA TRANSLATION_NAME
adSchemaTrustees	39	Reserved for future use.	
adSchemaUsagePrivileges	15	Returns the USAGE privileges on objects defined in the catalog that are available to, or granted by, a given user.	(USAGE_PRIVILEGES Rowset) OBJECT_CATALOG OBJECT_SCHEMA OBJECT_NAME OBJECT_TYPE GRANTOR GRANTEE
adSchemaViewColumnUsage	24	Returns the columns on which viewed tables, defined in the catalog and owned by a given user, are dependent.	(VIEW_COLUMN_USAGE Rowset) VIEW_CATALOG VIEW_SCHEMA VIEW_NAME
adSchemaViews	23	Returns the views defined in the catalog that are accessible to a given user.	(VIEWS Rowset) TABLE_CATALOG TABLE_SCHEMA TABLE_NAME
adSchemaViewTableUsage	25	Returns the tables on which viewed tables, defined in the catalog and owned by a given user, are dependent.	(VIEW_TABLE_USAGE Rowset) VIEW_CATALOG VIEW_SCHEMA VIEW_NAME

SearchDirectionEnum

This constant specifies the direction of a record search within a Recordset. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adSearchBackward	–1	Searches backward, stopping at the beginning of the Recordset. If a match is not found, the record pointer is positioned at BOF.
adSearchForward	1	Searches forward, stopping at the end of the Recordset. If a match is not found, the record pointer is positioned at EOF.

SeekEnum

This constant specifies the type of Seek to execute. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adSeekFirstEQ	1	Seeks the first key equal to KeyValues.
adSeekLastEQ	2	Seeks the last key equal to KeyValues.
adSeekAfterEQ	4	Seeks either a key equal to KeyValues or just after where that match would have occurred.
adSeekAfter	8	Seeks a key just after where a match with KeyValues would have occurred.
adSeekBeforeEQ	16	Seeks either a key equal to KeyValues or just before where that match would have occurred.
adSeekBefore	32	Seeks a key just before where a match with KeyValues would have occurred.

StreamOpenOptionsEnum

This constant specifies options for opening a Stream object. The values can be combined with an OR operation. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adOpenStreamAsync	1	Opens the Stream object in asynchronous mode.
adOpenStreamFromRecord	4	Identifies the contents of the Source parameter to be an already open Record object. The default behavior is to treat `Source` as a URL that points directly to a node in a tree structure. The default stream associated with that node is opened.
adOpenStreamUnspecified	–1	Default. Specifies opening the Stream object with default options.

StreamReadEnum

This constant specifies whether the whole stream or the next line should be read from a Stream object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adReadAll	–1	Default. Reads all bytes from the stream, from the current position onwards to the `EOS` marker. This is the only valid `StreamReadEnum` value with binary streams (Type is `adTypeBinary`).
adReadLine	–2	Reads the next line from the stream (designated by the `LineSeparator` property).

StreamTypeEnum

This constant specifies the type of data stored in a Stream object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adTypeBinary	1	Indicates binary data.
adTypeText	2	Default. Indicates text data, which is in the character set specified by `Charset`.

StreamWriteEnum

This constant specifies whether a line separator is appended to the string written to a Stream object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adWriteChar	0	Default. Writes the specified text string (specified by the Data parameter) to the Stream object.
adWriteLine	1	Writes a text string and a line separator character to a Stream object. If the `LineSeparator` property is not defined, then this returns a run-time error.

StringFormatEnum

This constant specifies the format when retrieving a Recordset as a string. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adClipString	2	Delimits rows by `RowDelimiter`, columns by `ColumnDelimiter`, and null values by `NullExpr`. These three parameters of the `GetString` method are valid only with a `StringFormat` of `adClipString`.

XactAttributeEnum

This constant specifies the transaction attributes of a Connection object. The constants are shown in the following table.

CONSTANT	VALUE	DESCRIPTION
adXactAbortRetaining	262144	Performs retaining aborts—that is, calling `RollbackTrans` automatically starts a new transaction. Not all providers support this.
adXactCommitRetaining	131072	Performs retaining commits—that is, calling `CommitTrans` automatically starts a new transaction. Not all providers support this.