

PROPERTIES

Main...

Technical Reference

[Document Conversions](#)

[IMEX Error Class](#)

[General Reference](#)

[Status Properties](#)

[Sql Statement](#)

Properties

[Action](#)

[Archive](#)

[Busy](#)

[Connect](#)

[Database](#)

[DatabaseName](#)

[DateFieldName](#)

[ErrorCount](#)

[Exclusive](#)

[FieldHeader](#)

[FileName](#)

[FieldSep](#)

[Header](#)

[HeaderLen](#)

[IniPath](#)

[LogFileNames](#)

[Name](#)

[Options](#)

[Password](#)

[QuoteChar](#)

[ReadOnly](#)

[Recordset](#)

[RecordsetType](#)

[RecordSource](#)

[RecordsProcessed](#)

[Status](#)

[SysEvents](#)

[TimeActive](#)

[UniqueIndex](#)

[UpdateOnDate](#)

[UpdateOnIndex](#)

[UseFieldHeader](#)

[UseUniqueHeader](#)

[UseQuotes](#)

[UserID](#)

[Version](#)

[Visible](#)

[About Method](#)

[Properties Method](#)

[ShowErrors Method](#)

Exclusive



Description Once set and an action applied to it, this Property instructs the IMEX Server to open the database exclusively, thus disallowing users to access the database while import/export actions are taking place.

Usage `IMEX.Exclusive = bVal`
`bVal = IMEX.Exclusive`

Data Type [Boolean](#)

Availability Read/Write

Comments This Property must be set before setting the '[Action](#)' Property. This Property gives increased performance as no record locking is required during import/export actions.

Options

INDEX...

Description The Options Property returns or sets a value that specifies one or more characteristics of the [Recordset](#) in the IMEX Server's Recordset.

Usage `IMEX.Options = iVal%`
`iVal% = IMEX.Options`

Data Type Integer

Availability Read/Write

Comments Use one or more of the following values to set the Options property. If you use more than one option, you must add their values:

Constant	Value	
dbDenyWrite	1	In a multi-user environment, other users can't make changes to records in the Recordset.
dbDenyRead	2	In a multi-user environment, other users can't read records (<i>table-type Recordset only</i>).
dbReadOnly	4	You can't make changes to records in the Recordset.
dbAppendOnly	8	You can add new records to the Recordset, but you can't read existing records.
dbInconsistent	16	Updates can apply to all fields of the Recordset, even if they violate the join/relational rules.
dbConsistent	32	(<i>Default</i>) Updates apply only to those fields that don't violate the join/relational rules.
dbSQLPassThrough	64	When using an SQL statement in the RecordSource property, sends the SQL statement to an ODBC database, such as a SQL Server or Oracle database, for processing.
dbSeeChanges	512	Generate a trappable error if another user is changing data you are importing with the IMEX Ole Server.

To set more than one value for this property, you can combine options by adding values together. For example, to set both dbReadOnly and dbInconsistent you can use this code:

```
IMEX.Options = dbAppendOnly + dbInconsistent
```

To determine if the property contains a specific value, you can use the And operator. For example, to find out if the Recordset is open for read only access, you could use this code:

If IMEX.Options And dbReadOnly Then...

Using both dbInconsistent and dbConsistent results in consistent updates, the default for the IMEX Server's Recordset object.

Action

INDEX...

Description The Action Property starts the specific action.

These actions can be:

- 0 No Action
- 1 Action Import
- 2 Action Export
- 3 Reserved
- 4 Reserved
- 5 Reserved
- 6 Reserved
- 7 Reads unique header & will auto set properties for you

Usage `IMEX.Action = IntegerAction%`
`IntegerAction% = IMEX.Action`

Data Type Integer

Availability Read/Write

Comments Before any Action can take place other related properties **must** have been set

Busy

INDEX...

Description This Property discloses the current status of the IMEX Ole Server.

Usage `bVal = IMEX.Busy`

Data Type [Boolean](#)

Availability Read Only

Comments This status property should be checked prior to setting any action because a trappable error will occur when trying to set the Action property, when the Server is Busy.

Connect

INDEX...

Description Returns or sets a value that provides information about the source of an open database, a database used in a pass-through query, or an attached table.

Usage IMEX.Connect = [*databasetype*;*parameters*;]

Data Type String

Availability Write only

Comments *databasetype*
A string expression that specifies a database type. For Jet databases, exclude this argument; if you specify parameters, use a semicolon (;) as a placeholder. (*Data type is String.*)

parameters
A string expression that specifies additional parameters to pass to ODBC or installable ISAM drivers. Use semicolons to separate parameters. (*Data type is String.*)

Settings The Connect property setting is a String composed of a database type specifier and zero or more parameters separated by semi-colons. The Connect property is used to pass additional information to ODBC and certain ISAM drivers as needed. It isn't used for Jet databases, except for those containing attached tables, so as to allow SQL pass-through queries. To perform an SQL pass-through query on a table attached to your .MDB file, you must first set the Connect property of the attached table's database to a valid ODBC connect string.

For a TableDef object that represents an attached table, the Connect property setting consists of one or two parts (*a database type specifier and a path to the database*), each of which ends with a semi-colon.

The path as shown in the table below is the full path for the directory containing the database files and must be preceded by the identifier "DATABASE=". In some cases (*as with Jet, Btrieve, and Microsoft Excel databases*) a specific filename is included in the database path argument.

The following table shows possible database types and their corresponding database specifiers and paths for the Connect property setting:

Database type	Specifier	Path
Database using the Jet database engine		drive:\path\filename.MDB
dBASE III	dBASE III	drive:\path
dBASE IV	dBASE IV	drive:\path
Paradox 3.x	Paradox 3.x	drive:\path
Paradox 4.x	Paradox 4.x	drive:\path
Btrieve	Btrieve	drive:\path\filename.DDF
FoxPro 2.0	FoxPro 2.0	drive:\path
FoxPro 2.5	FoxPro 2.5	drive:\path

FoxPro 2.6	FoxPro 2.6	drive:\path
Excel 3.0	Excel 3.0	drive:\path\filename.XLS
Excel 4.0	Excel 4.0	drive:\path\filename.XLS
Excel 5.0	Excel 5.0	drive:\path\filename.XLS
Text	Text	drive:\path
ODBC	*(See Below)	None

*DATABASE=defaultdatabase; DATABASE=defaultdatabase; UID=user; PWD=password;
DSN=datasourcename; LOGINTIMEOUT=seconds
(This may not be a complete connection string for all servers; it's just an example.)

If the specifier is only "ODBC;", a dialog box listing all registered ODBC data source names is displayed by the ODBC driver so the user can select a database. If a password is required but not provided in the Connect property setting, a login dialog box is displayed the first time a table is accessed by the ODBC driver and again if the connection is closed and reopened. For Jet database base tables, the Connect property setting is a zero-length string ("").

Please refer to Microsoft Visual Tools documentation relating to ODBC.

Database

INDEX...

Description This Property serves as access to the logical database object after the 'Action' property has been set.

Usage Set *Object* = IMEX.Database

Data Type Database Object

Availability Read/Write (*During Import/Export Actions only*)

Comments Please note that we do not recommend setting dependant objects to the Server's database object as it is transient. Due to the fact that the database is opened at the start of any action and closed once the action has completed or failed. Hence this causes unpredictable results.

DatabaseName

INDEX...

Description Returns or sets the name and location of the source of data for the IMEX Server.

Usage IMEX.DatabaseName = *pathname*
pathname = IMEX.DatabaseName

Availability Read/Write

Data Type String

Comments If your network system supports it, the *pathname* argument can be a fully qualified network path name such as \\MYSERVER\MYSHARE\DATABASE.MDB. The database type is indicated by the file or directory that *pathname* points to, as follows:

<i>pathname</i> points to...	Database Type
.MDB file	Microsoft Access database
.DDF file	Btrieve database
Directory containing .DBF file(s)	dBASE database
Directory containing .XLS file	Microsoft Excel database
Directory containing .DBF files(s)	FoxPro database
Directory containing .PDX file(s)	Paradox database
Directory containing appropriate database files	Text format database

ErrorCount

INDEX...

Description The ErrorCount Property is the total accumulation of all errors for the current action being processed

Usage `!val% = IMEX.ErrorCount`

Data Type Integer

Availability Read only

Comments A count of the IMEX error collection, see also [ShowErrors](#) method

FieldHeader

INDEX...

Description The FieldHeader Property is a container for field names (*columns*) for the import/export text. It conforms to the Microsoft Access Import/Export standard for field name headers.

Usage `String$ = IMEX.FieldHeader`
`IMEX.FieldHeader = String$`

Data Type String

Availability Read only

Comments This property is especially useful when importing a file where the fields (*columns*) of a [Recordset](#) are in an unknown order as the IMEX Server automatically matches the source and target field names. This can be set manually for specific field headers or allow IMEX Server to create this automatically.

We recommend you use the FieldHeader Property as it gives the information to the IMEX Ole Server so that fields (*columns*) can be arranged independant to the order held within the Recordset. The IMEX Ole Server has the functionality to maintain a discipline between target and source fields (*columns*) hence you are free to ommit or have a surplus of fields (*columns*) within either the target or the source. When a field is found in both target and source they will automatically be matched together and the data updated.

Related topics: [UpdateOnDate](#)
[UpdateOnIndex](#)
[UseFieldHeader](#)

FileName

INDEX...

Description The FileName Property is a fully qualified network or local path to a file.

Usage IMEX.FileName = *String\$*
String\$ = IMEX.FieldName

Data Type UNC String

Availability Read/Write

Comments To preserve file integrity this property is cleared by the IMEX Server after any action so that files are not accidentally overwritten. This this functionality is included as 'Ole Server' rulings discourage the use of warning dialogs thus enabling the IMEX Server to run in a fully automated manner (i.e. no user intervention).

On export this property can be left blank and a temporary filename will automatically be generated by the IMEX Server in the Windows temp directory, if this Property is unused on export the FileName Property **must** be read shortly after setting the 'Action' property to export due to the automatic clearing of file names after action completion.

NB The 16 bit version cannot support long filenames (*see Registration for how to obtain the 32 bit versions*).

FieldSep

INDEX...

Description The FieldSep Property is an ASCII integer value, this value is the character code used for delimiting fields in import/export text [i.e Chr\$ (44)].

Usage IMEX.FieldSep = *iVal*
iVal = IMEX.FieldSep

Data Type Integer

Availability Read/Write

Comments These standard ASCII character codes can be one of the following:

ASCII 9	{Tab}	
ASCII 32	{Space}	
ASCII 44	{Comma}	Default
ASCII 59	{Semi-colon}	Dutch/Eurpoean

Header

INDEX...

Description The Header is a 'luggage placeholder', in other words it is a user/programme defined header consisting of any valid ASCII characters up to 64k in length.

Usage IMEX.Header = *String\$*
String\$ = IMEX.Header

Data Type String

Availability Read/Write

Comments Although it may contain many carriage return and line feeds it must be terminated with an ASCII carriage return and line feed (*ASCII 12 & 10*). In the case of missing termination the IMEX Server will automatically append the necessary ASCII termination characters.

See related Property: [HeaderLen](#)

HeaderLen

INDEX...

Description The HeaderLen Property is the length of the [Header](#).

Usage `!Val& = IMEX.HeaderLen`

Data Type Long

Availability Read only (*after Header Property is set*)

Comments Used for disassembling or interpreting the 'luggage'. In other words assists the programmer by offering a pointer that can indicated the true 'BOF'.

ReadOnly

INDEX...

Description The ReadOnly Property, as the name implies provides a read only database. Therefore not only the table or Recordset is read only but the entire database and attached tables where the database allows..

Usage `IMEX.ReadOnly = bVal`
`bVal = IMEX.ReadOnly`

Data Type Boolean

Availability Read/Write

Comments Use this Property to protect against accidental writes to the database. A fail-safe Property, we would generally recommended for debugging your programmes and testing the IMEX Ole Server

Recordset

INDEX...

Description The [Recordset](#) Property returns or sets a Recordset object within the IMEX Server's own Recordset.

Usage Set *Object* = IMEX Recordset

Data Type [Variant](#)

Availability Read only

Comments Please note that we do not recommend setting dependant objects to the Server's Recordset object as it is [transient](#). Due to the fact that the Recordset is created at the start of any action and closed once the action has completed or failed. Hence this causes unpredictable results.

RecordsetType

INDEX...

Description The RecordsetType Property Returns or sets a value indicating the type of [Recordset](#) object you want the IMEX Server to create.

Usage IMEX.RecordsetType = *iVal%*
iVal% = IMEX.RecordsetType

Data Type Integer

Availability Read/Write

Comments This can be a constant or value that specifies a type of Recordset, as described in settings listed below. Generally when importing a Dynaset or Table-type of Recordset should be requested as the IMEX Server needs to write to the Recordset. On exporting a snapshot type of Recordset should be selected as this provides a performance increase.

However if you are using the [UpdateOnImport](#) Property to update/add records on import then you **must** select a Recordset of the Table-type, because an indexed import requires a [UniqueIndex](#) to perform the high speed lookups required for this operation.

Settings	Value	Description
vbRSTypeTable	0	A table-type Recordset.
vbRSTypeDynaset	1 (Default)	A Dynaset-type Recordset
vbRSTypeSnapshot	2	A snapshot-type Recordset.

If the IMEX Server cannot create the type of Recordset you requested, an error occurs. If you don't specify a RecordsetType before the IMEX Server creates the Recordset, a dynaset-type Recordset is created (*if possible*) automatically for you.

RecordSource

INDEX...

Description The RecordSource Property returns or sets the underlying source for creating the IMEX Server's [Recordset](#). This can be an SQL statement, the name of a Query or the name of a table within the current database.

Usage IMEX.RecordSource = *String\$*
String\$ = IMEX.RecordSource

Data Type String

Availability Read/Write

Comments A string expression specifying a name, as described in Settings.
The settings for value are:

Setting Description

A table name The name of one of the tables defined in the Database.

An SQL query A valid SQL string using syntax appropriate for the database.

A Query The name of one of the Queries defined in the Database.

The RecordSource property specifies the source of the records accessible through the IMEX Ole Server, which in turn make up the [Recordset](#). If you set the RecordSource property to the name of an existing table in the database, all of the fields in that table are accessible for import and export operations. The order of the records retrieved is determined by an 'Order By' clause in the RecordSource select statement. If you don't specify the 'Order By' clause, the data is returned in no particular order.

If you set the RecordSource property to the name of an existing Query in the database, all fields returned by the Query are also made available for import/export operations by the IMEX Server. The order of the records retrieved is set by the Query, if the Query doesn't specify an order, the data, again is returned in no particular order.

If you set the RecordSource property to an SQL statement it **must** be a statement that returns records, all fields returned by the SQL query are made available to the IMEX Server. This statement may include an 'Order By' clause to change the order of the records returned by the Recordset created by the IMEX Server or a 'Where' clause to filter the records. If the database you specify in the [Database](#) and [Connect](#) property isn't a Microsoft Jet engine database, or if the [dbSQLPassThrough](#) option is set in the Options property, your SQL query must use the syntax required by that database engine.

N.B. Whenever your Query or SQL statement returns a value from an expression and not an actual field name a name is created automatically by the IMEX Ole Server. Generally, the name is Expr1 followed by a three character number beginning with 000. For example, the first expression would be named: Expr1000.

In most cases you will want to alias expressions so you know the name of the column that will be produced. See the [SQL SELECT statement AS](#) clause for more

information.

RecordsProcessed

INDEX...

Description The RecordsProcessed Property is an indicator that during import/export operations states how many records have been imported or exported at that particular snapshot in time.

Usage `!Val! = IMEX.RecordsProcessed`

Data Type Long

Availability Read only

Comments This property helps you monitor the status of the current operation, it is also useful to compare records processed to 'ErrorCount' and assess whether the operation is returning an acceptable rate of success. Possible even set an automatic abort when the error percentage exceeds what you can accept.

Status

INDEX...

Description The Status Property returns the current percentage status of the action being currently performed.

Usage `iVal% = IMEX.Status`

Data Type Integer

Availability Read only

Comments Returns the percentage complete, a useful when calculating a success rate between ErrorCount, RecordsProcessed and Status. Saving the entire process being performed if there are fundamental problems during import or export that would be detrimental or prove to be a waste of time and resources.

UseFieldHeader

INDEX...

Description The UseFieldHeader specifies that a field header should be constructed and prefixed to the file.

Usage `bVal = IMEX.UseFieldHeader`
`IMEX.UseFieldHeader = bVal`

Data Type [Boolean](#)

Availability Read/Write

Comments See [FieldHeaderProperty](#)

We recommend you use the UseFieldHeader Property as it gives the information to the IMEX Ole Server so that fields (*columns*) can be arranged independantly of the order held within the [Recordset](#). The IMEX Ole Server has the functionality to maintain a discipline between target and sourde fields (*columns*) hence you are free to ommit or have a surplus of fields (*columns*) within either the target or the source. When a field is found in both target and source they will automatically be matched together and the data updated.

UseQuotes

INDEX...

Description The UseQuotes Property indicates whether text delimiters are to be used (*normally quotation marks ASCII 34*).

Usage $bVal = \text{IMEX.UseQuotes}$
 $\text{IMEX.UseQuotes} = bVal$

Data Type [Boolean](#)

Availability Read/Write

Comments Default for this Property is TRUE.
See: ['QuoteChar'](#)

Version



Description The Version Property returns the current major, minor and build version for the IMEX Ole Server.

Usage `String$ = IMEX.Version`

Data Type String

Availability Read only

Comments Use this Property for checking the current version of the control to maintain compatibility. We would recommend for bulletproof applications that you check this Property before setting any other Properties or setting any actions.

General Reference

INDEX...

Technical Overview

The IMEX Server is automatically initialized when your application creates a reference to the Server.

At run-time, after the Server has successfully been initialised the following properties must be set before any Action is started:-

[\[Connect\]](#)

[DatabaseName](#)

[\[Options\]](#)

[RecordSource](#)

[\[Exclusive\]](#)

[\[ReadOnly\]](#)

[\[RecordsetType\]](#)

(Properties in "[]" are optional.)

The IMEX Server's database engine (*MSJET*) attempts to create a new [Recordset](#) object internally based on the property settings. This Recordset is accessible through the IMEX Server's [Recordset](#) property.

You can use the Recordset property as you would any other Recordset object. For example, you can use any of the Recordset methods or properties and examine the structure of the Recordset object's underlying schema.

Note: The RecordSet Property is transient and therefore it is not recommended to set dependant objects to the RecordSet Property.

You can also request the type of Recordset to be created by setting the IMEX Server's [RecordsetType](#) property. If you don't request a specific type, a dynaset-type Recordset is created automatically by the IMEX Ole Server. Using the RecordsetType property, you can request to create either a table-, snapshot or dynaset-type Recordset. However, if the IMEX database engine (*MSJET*) can not create the type requested, an error occurs.

The type of Recordset created can be determined at run time by examining the IMEX's RecordsetType property.

For example: If IMEX.RecordsetType = 1 Then ...
If IMEX.RecordsetType = RSTypeTable Then ...

ShowErrors

INDEX...

Description The ShowErrors Method returns a variant which itself contains a two dimensional variant array.
The 'IMEX Error Class' is a collection which makes the unattended functionality of the 'IMEX Ole Import/Export' possible it even removes the need for **any** error trapping code!

Usage `vArray = IMEX.ShowErrors`

Data Type Variant Array

Availability Read only

Comments After the array is returned the 'IMEX Error Collection's cleared and the error count is reset to zero.

Array Contents `vArray(x%,y%)` - values within the array

Dimension x% = the unique error event number related to the 'ErrorCount' at that snapshot in time.

Dimension y% = consists of four variants:-

- 0 Error number (*long*)
- 1 Error description (*string*)
- 2 Error source (*string*)
- 3 Error action (*string*)

Number A long value (*contained in a variant*) which indicates an error number, this number is offset (*overloaded*) by a standard Microsoft Ole object error (`vbObjectError`). To arrive at a real error number you must subtract `vbObjectError` from the error number.

Description This is quite simply the explanation of the error in English. On import and export this will explain what line was in error and for what reason.

Source The source of the error (*mainly used internally by IMEX Server*).

Action The action is what was being processed at the time of the error.

Luggage

From time to time in various applications, often relating to communications, Header structures are prefixed to the actual data section of the file.

These can be useful in the following scenarios:

The IMEXComms Ole Server uses these luggage placeholders for file routing and computer/programme intercommunications. This enables any remote instance of the application to make quasi-decisions based upon the content of the luggage. In essence this is either a request or a response to a previous request from which the IMEXComms Ole Server can successfully inform the requester that the specific request has been fulfilled or indeed failed.

The nature of this luggage or header structure lends itself to a multitude of uses from date time stamping to encryption keys for secure file transfer. Fundamentally, whatever your imagination allows or requirements demand can be accommodated by this concept.

UpdateOnIndex



Description The UpdateOnIndex Property indicates whether the target record should be updated as opposed to added based upon the [UniqueIndex](#) Property.

Usage *bVal* = IMEX.UpdateOnIndex
IMEX.UpdateOnIndex = *bVal*

Data type [Boolean](#)

Availability Read/Write

Comments See [UniqueIndex](#) Property

UniqueIndex

INDEX...

Description The UniqueIndex Property is used to implement a unique way of either importing or updating to existing records from an external source.

Usage `String$ = IMEX.UniqueIndex`
`IMEX.UniqueIndex = String$`

Data Type String

Availability Read/Write

Comments Used in conjunction with a Table-type [Recordset](#) and [UpdateOnImport](#) Property on an import action this property offers a checked import.

For example, the data from each record to be imported is checked against the index set in UniqueIndex Property and if it finds a target record in the [Recordset](#) the source record is updated into that target record. However if the lookup fails the record is imported or added to the Recordset.

NB The unique index passed to the UniqueIndex Property can have multiple fields and the IMEX Ole Server automatically resolves the data lookup for each field. If you have the requirement we genuinely recommend you try this, although it may appear to incur an unwieldy overhead it performs very well. Depending on your selection of indexes this Property can has the ability to save a great deal of de-duplication after import.

This Property is currently being enhanced and will feature an integral time/date check plus other enhancements for record comparison imports.

SQL statement

INDEX...

Introduction It is recommended that you use an SQL Statement set into the [RecordSource](#) Property for **all** exports. By doing this you produce the fastest possible means of exporting data and avoid having to pre-build [RecordSet](#) objects which are then passed to an Ole Server and become dependant objects. If you were to use this method you would severely restricts the possible speed advantages that the 'IMEX Ole Server' is designed to offer.

In addition we can avoid delving into the mirky depths of OLE automation inter-communications **on** dependant Recordset objects, although Ole Inter-Communications is generally very fast, by comparison using an SQL Select Statement our tests have proved this to be in excess of 110 times faster than a dependant Recordset object. The speed difference is often likened to using an Commodore Pet processing Microsoft's international annual accounts. (*We are talking S L O O W W W!*)

On a more serious note using an SQL Select isolates any database actions (*by encompassing all the database functionality*), your program or development environment does not even have to support SQL or Recordset objects.

Exporting Data Using SQL

To learn more about the parts of an SQL statement, read the respective section in your chosen database's manual or if your database language does not reference SQL there are a range of information sources on both CompuServe and Internet Web Sites. For Microsoft Visual Tools users search Help for any of the SQL reserved words listed below:

```
SELECT fieldlist
FROM tablenames IN databaseName
WHERE searchconditions
GROUP BY fieldlist
HAVING searchconditions
ORDER BY fieldlist
```

To use an SQL statement with the [RecordSource](#) property in your program, build a string containing the statement. Then either include it as a literal quoted argument, or as a variable in a method, statement, or function.

Because SQL statements can be very long, you may wish to break them up to simplify and understand the SQL Statement you have created.

SQL statements can not contain embedded carriage return characters, because these act as statement terminators.

SELECT Statement (SQL)

Instructs the IMEX Ole Server to obtain it's export information from the database as a set of records.

Syntax

```
SELECT [predicate] { * | table.* | [table.]field1 [, [table.]field2[, ...]] }
[AS alias1 [, alias2 [, ...]]]
FROM tableexpression [, ...] [IN externaldatabase]
[WHERE... ]
```

[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

The SELECT statement has these parts:

Part	Description
predicate	<i>One of the following predicates: ALL, DISTINCT, DISTINCTROW, or TOP. You use the predicate to restrict the number of records returned.</i>
*	<i>Specifies all fields from the specified table(s) are selected.</i>
table	<i>The name of the table containing the fields from which records are selected (database files).</i>
field1, field2	<i>The names of the fields to retrieve data from. If you include more than one field, they are retrieved in the order listed.</i>
alias1, alias2	<i>The names to use as column headers (filed names) instead of the original column names in the table.</i>
tableexpression	<i>The name of the table or tables containing the data you want to retrieve.</i>
externaldatabase	<i>The name of the database containing the tables in tableexpression if not in the current database.</i>

Comments To perform this operation, the chosen IMEX database engine (*MSJET*) searches the specified table(s), extracts the chosen columns (*fields*), selects rows that meet the criterion, sorts and/or groups the resulting rows into the order specified. **SELECT** statements don't change data in the database.

SELECT is usually the first word in an SQL statement. Most SQL statements are either **SELECT** or **SELECT...INTO** statements.

The minimum syntax for a **SELECT** statement is:

```
SELECT fields FROM table
```

You can use an asterisk (*) to select all fields in a table. The following example selects all of the fields from an Employees table:

```
SELECT * FROM Employees;
```

If a field name is included in more than one table in the FROM clause, precede it with the table name and the . (dot) operator. In the following example, the Department field is in both the Employees table and the Supervisors table. The SQL statement selects Department from the Employees table and SupvName from the Supervisors table:

```
SELECT Employees.Department, SupvName  
FROM Supervisors, Employees  
WHERE Employees.Department = Supervisors.Department;
```

Use the **AS** reserved word in order to give an alias (*alternative name*). The following

example uses the title [Birth Date] to name the returned Field in the resulting Recordset:

```
SELECT [Birth Date]  
AS Birth FROM Employees;
```

Please note that the square brackets [] in this instance are used to encompass a field name with an embedded space, this is contrary to the Document Conventions.

You can use the other clauses in a SELECT statement to further restrict and organize your returned data.

Example of Select statements

```
SELECT LastName, FirstName FROM Employees  
Selects the LastName and FirstName fields of all records in the Employees table.
```

```
SELECT Employees.* FROM Employees  
Selects all fields from the Employees table.
```

Information from 'Microsoft Visual Tools, books on line.

QuoteChar

INDEX...

Description The QuoteChar Property holds the ASCII equivalent character code for the text delimiter (*Default = ASCII 34 - double quotes*).

Usage $iVal\% = \text{IMEX.QuoteChar}$
 $\text{IMEX.QuoteChar} = iVal\%$

Data Type Integer

Availability Read/Write

Comments Used in conjunction with the [UseQuotes](#) Property and is a digit placeholder for one of the following text delimiters.

ASCII 0	{Null}	No quotes
ASCII 34	{Double Quotes}	Default
ASCII 39	{Single Quote}	

Password

INDEX...

Description The Password Property is used in conjunction with the [UserID](#) Property for connecting to secure JET databases only.

Usage IMEX.Password = *String*\$

Data Type String

Availability Write only

Comments This is only set once prior to database initialisation, this should not be confused with ODBC UserID and password settings as these are set with the [Connect](#) Property

LogFileName

INDEX...

Description The LogFileName Property is a fully qualified Path and File Name, and when set points to a log file where session and error status will be recorded.

Usage *String\$* = IMEX.LogFileName
IMEX.LogFileName = *String\$*

Data Type UNC String

Availability Read/Write

Comments This Property can be left uninitialised if you do not want to have a log file maintained by the IMEX Ole Server.

IniPath

INDEX...

Description The IniPath Property sets the location and file name of the initialisation file to be used to set the options for the database engine.

Usage `String$ = IMEX.IniPath`
`IMEX.IniPath = String$`

Data Type String

Availability Read/Write

Comments Used only once, this Property initialises the Jet database engine and holds a path normally to your own application's '.ini' file. Please see the [example](#) for the default settings for Jet and ODBC settings.

About Method

INDEX...

Description The About Method displays Registration and version information within an 'About' dialog box.

Usage IMEX.About

Return Val None or void

Comments When the product is registered this method becomes redundant as the 'About' version information can be attained programmatically from the [Version](#) Property. Also note that the file size decreases by about 100k thus producing a smaller overhead in the registered copy.

Archive

INDEX...

Description The Archive Property deletes records directly after export has occurred.

Usage *bVal* = IMEX.Archive
IMEX.Archive = *bVal*

Data Type [Boolean](#)

Availability Read/Write

Comments Only available for export

Visible

INDEX...

Description The Visible Property displays a small tag in the top right corner of the screen, showing the current status of any import/export function taking place.

Usage *bVal* = IMEX.Visible
IMEX.Visible = *bVal*

Data Type [Boolean](#)

Availability Read/Write

Comments Use this Property to display to the user of the application a [percentage complete](#) and [Records Processed](#).

UserID

INDEX...

Description The UserID Property is used in conjunction with the [Password](#) Property for connecting to secure JET databases only.

Usage IMEX.UserID = *String\$*
String\$ = IMEX.UserID

Data Type String

Availability Read/Write

Comments This is only set once prior to database initialisation, this should not be confused with ODBC UserID and password settings as these are set with the [Connect](#) Property

Properties Method

INDEX...

Description The Properties Method is a global snapshot of the current internal status values of **all** the status type properties. (See [Status Properties](#))

Usage `vVar = IMEX.Properties`

Return Val. [Variant \(Array\)](#) alias 'ANY' data type

Comments We would recommend that you look at the functionality within the demo application to gauge the uses as it is best understood from a real example.

Fundamentally this Method throws out all the [status properties](#) from the Server in one 'lump!' The code in the demo application is all done and please feel free to use this and modify this for your own use, the code is supplied 'as is' and without warranties. As you can see from the code, this Method's return value upon inspection generates the controls automatically from the returned data types held within the variant. This is particularly useful if you want a self maintaining, full status display within you application. The controls automatically add themselves if a new Property is introduced, producing an enhancement proof area to the status display within your distributed applications.

Array Contents `vArray(x%,y%)` - values within the array

Dimension x% is a unique number identifying the position of the returned value in the array (0-25)

Dimension y% consists of three variants, namely:-

- 0 Property Name (*string*)
- 1 Data type (*held within 'Returned Value'*)
- 2 Returned value (*see... [Returned VariableTypes](#)*)

Property Name A string containing (*contained in a variant*) which is the actual Property Name that is being returned within this segment of the array.

Data Type The actual data type that the data was before being converted to a variant by this Method. All these values are integers from 0-17, for a description of these values see [Returned Variable Types](#).

Returned Value This is the actual status values, the data type for these values is held within the 'Data Type' segment of the array.

Ini File Example

Settings taken from 'biblio.ini' of the Demo Application

Below is an example from an MS Access 2.5 initialisation file, please refer to your own database documentation for the specific default settings that your own database engine requires.

[Options]

SystemDB=C:\MSACCESS\SYSTEM.MDA ;Path and File Name for security database

[ISAM]

;Database engine defaults

PageTimeout=5

LockedPageTimeout=5

CursorTimeout=10

LockRetry=20

CommitLockRetry=20

MaxBufferSize=512

ReadAheadPages=16

IdleFrequency=10

[ODBC]

;ODBC Database engine defaults

QueryTimeout=60

LoginTimeout=20

[Language]

;This sets the sort/locale settings

Language=GENERAL

Name

INDEX...

Description The Name Property is a unique identifier (*Name*) for the running instance of the IMEX Ole Server

Usage String\$ = IMEX.Name
IMEX.Name = String\$

Data Type String

Availability Read/Write

Comments Initially a unique name is defined by the IMEX Ole Server, this is made up of the '**NAME**' (i.e. '*Import/Export OLE Server*') and the '**hInst**' which is a windows generated unique handle given to the specific running instance. This can be a unique string of your own choosing, however we recommend that each '**NAME**' is unique for hopefully obvious reasons.

Example Instance 'One' of two running instances of the IMEX Ole Server could be importing to a large corporate database into a table called 'Orders', where as instance 'TWO' could be importing in the same large database into a table called 'Sales'. Therefore...

Instance 'One' could be called - '**Importing/Orders**'
Instance 'Two' could be given the Name Property '**Importing/Sales**'.

In situations where exporting is involved the example above shows the relevance of setting a unique Name Property to describe the specific instance of the IMEX Ole Server.

Status Properties

INDEX...

Below is a list of the Property Values that are returned by the [Properties Method](#)

Action
Archive
Busy
DateFieldName
ErrorCount
Exclusive
FieldHeader
FieldSep
FileName
Header
HeaderLen
LogFileName
Name
QuoteChar
ReadOnly
RecordsProcessed
Status
SysEvents
TimeActive
UniqueIndex
UpdateOnDate
UpdateOnIndex
UseFieldHeader
UseQuotes
UseUniqueHeader
Version
Visible

UseUniqueHeader

INDEX...

Description The UseUniqueHeader Property date and time stamps the individual export file and also adds a automatic features that aid recognition if this files is to be imported at a later date.

Usage *bVal* = IMEX.UseUniqueHeader
IMEX.UseUniqueHeader = *bVal*

Data Type [Boolean](#)

Availability Read/Write

Comments The features that are added to the file via means of a unique header are the following:

- Date/time stamp
- Automatic field delimiter recognition
- Automatic text delimiter recognition
- Automatic user defined header recognition

TimeActive

INDEX...

Description The TimeActive Property returns a variant which holds the amount of clock ticks that a particular import/export action has taken.

Usage `vVal = IMEX.TimeActive`

Data Type Variant

Availability Read only

Comments This Property indicates the amount of time taken and as such returns a status for your own applications.

SysEvents

INDEX...

Description The SysEvents Property, when set (*TRUE*) activates the IMEX Ole Server as a time sharing Server that shares time resources during import and export actions so that the system can receive time to process any concurrent background processes.

Usage bVal = IMEX.SysEvents
IMEX.SysEvents = bVal

Data Type Boolean (*TRUE - default*)

Availability Read/Write

Comments Set this Property to FALSE (0) if you want to suspend concurrent tasks when importing or exporting records. This can be especially useful if you have a requirement where after importing/exporting records further actions need to be immediately carried out on that specific data, without interruption.

UpdateOnDate

INDEX...

Description The UpdateOnDate Property is a record Date/Time comparison which dictates whether the source or indeed the target record is the most recent before being updated.

Usage *bVal* = IMEX.UpdateOnDate
IMEX.UpdateOnDate = *bVal*

Data type [Boolean](#)

Availability Read/Write

Comments This should only be used in conjunction with the [UniqueIndex](#), [UpdateOnIndex](#) and [UpdateOnDate](#) Properties

Returned Variable Types

Value	Variable type description
0	Empty (uninitialized).
1	Null (no valid data).
2	Integer.
3	Long integer.
4	Single-precision floating-point number.
5	Double-precision floating-point number.
6	Currency.
7	Date.
8	String.
9	OLE Automation object.
10	Error.
11	<u>Boolean</u> .
12	<u>Variant</u> (used only with arrays of Variants).
13	Non-OLE Automation object.
17	Byte

DateFieldName

INDEX...

Description The DateFieldName Property is the field name that will be compared with the Date/Time comparison which dictates whether the source or indeed the target record is the most recent before being updated.

Usage `String$ = IMEX.DateFieldName`
`IMEX.DateFieldName = String$`

Data type String

Availability Read/Write

Comments This should only be used in conjunction with the [UniqueIndex](#), [UpdateOnIndex](#) and [UpdateOnDate](#) Properties.
For this property to work correctly the specific 'date' field name **must** exist within the [Recordset](#) and valid data for this field **must** exist in the data import file.

IMEX Document Conventions

INDEX...

Description This section explains the document layout conventions used within IMEX documentation to aid understanding, hopefully avoiding ambiguities and lengthy explanations.

Code Conventions

Parenthesis are used to explain relationships in code, these are formatted as below.

[Expression List] *In syntax, items inside square brackets are optional*

{WHILE | UNTIL} *In syntax, braces and a vertical bar indicate a mandatory choice between two or more items. You must choose one of the items unless all of the items are enclosed in square brackets [].*

For example [{THIS | ORTHAT}]

Colour Conventions


Throughout the documentation colours are used (*in Help file only*) to separate **expressions** from **explanations**, (*notes*) and **labels**.

The help file itself shows:

Pop-Ups as - Pop-up

Jumps as - Jump

Jump

This is a Jump, click the 'back' button to return to the previous Topic or the  button for returning to the Properties Index.

Glossary



A

Array

B

BOF

Boolean

C

Clock Tick

P

Pop-up

R

Recordset

T

transient

U

UNC

V

Variant

Array

A tightly grouped multi - dimensional variable, which can be referenced as a single entity or can be indexed by a numeric off-set, to obtain just one of it's internal values.

BOF

Beggining Of File

Boolean

-1 (True)

0 (False)

Clock Tick

Internal timing measurement used in Windows.

1,000 Clock Ticks is approx. 1 Second

Pop-up

This is a Pop-Up, click here or press any key to return to the main topic.

Recordset

There are three types of Recordset, namely:

Table

The table type Recordset works directly over the database table/database file.

Dynaset

The Dynaset type of Recordset is a dynamic set of records held within memory much of the functionality is common in both Dynaset and Table type recordsets, except that the Table type recordset has a faster indexed lookup method (seek).

Snapshot.

The Snapshot has much of the same functionality as a Dynaset however a Snapshot is a fixed set of records held in memory that can not be written to.

transient

Only valid for a particular slice of time.

UNC

Universal Naming Conventions for network/filenames

Variant

Special type of variable that is is synonomous with Visual Basic and can contain any other Data Type.
i.e. String, Long, Integer, Boolean etc..

