GetRows

The *GetRows* method retrieves multiple rows from a **Recordset** object.

tags: Methods

Applies to ...

Object	Description
Recordset	A set of records derived from a table, a query or a SQL statement.

Syntax

recordset`.GetRows(`*numrows*`)`

Argument	Optional	Type	Description	Returned value
recordset		**Recordset object**	An open recordset	*GetRows* returns a two-dimensional array. The first subscript identifies the field and the second identifies the row number.
numrows	N	Long	Specifies the maximum number of rows to retrieve.	

Remarks

GetRows returns a two-dimensional zero-based array.
For example, to get the first field value in the second row returned, use code like the following:

```
Dim vVarRecords() As Variant, oRecordset As Object, vField1 As Variant
        Set oRecordset = ... OpenRecordset(...)
        Set vVarRecords = oRecordset.GetRows(1000)
        vField1 = vVarRecords(0, 1).
```

To get the second field value in the first row, use code like the following:

```
Dim vField2 As Variant
        vField2 = vVarRecords(1, 0)
```

The `vVarRecords` variable automatically becomes a two-dimensional array when *GetRows* returns data. Otherwise *GetRows* will return an uninitialized array (i.e. the output of the Basic builtin `Array()` function).
Each element of the returned array will be of the Basic type most close to the **DataType** of the original database field.
A `Null` field value in the database will be returned as a `Null` value in the array resulting from the invocation of *GetRows*. To be evaluated with the `IsNull()` Basic builtin function.

If you request more rows than are available, then *GetRows* returns only the number of available rows. You can use the Basic `UBound` function to determine how many rows *GetRows* actually retrieved, because the array is sized to fit the number of returned rows.
For example, you could use the following code to determine how many rows were actually returned:

```
        numReturned = UBound(vVarRecords, 2) + 1
```

You need to use "+ 1" because the first row returned is in the 0th element of the array. The number of rows that you can retrieve is constrained by the amount of available memory. You shouldn't use *GetRows* to retrieve an entire table into an array if it is large.
Because **GetRows** returns all fields of the *Recordset* into the array, including Memo fields, you might want to use a query that restricts the fields returned.

Binary fields are not returned. Instead the corresponding element in the array returned by *GetRows* contains the *length* of the field.

After you call *GetRows*, the current record is positioned at the next unread row. That is, *GetRows* has the same effect on the current record as `recordset.Move(numrows)`.
If you are trying to retrieve all the rows by using multiple *GetRows* calls, use the **EOF** property to be sure that you're at the end of the *Recordset*. *GetRows* returns less than the number requested if it's at the end of the *Recordset*.

Calling *GetRows* cancels all not confirmed updates on the *recordset*.

Error messages

Argument nr.1 (Value='...') is invalid
Recordset has been closed. Recordset action rejected

See also

BOF, EOF
Move (recordset)
OpenRecordset

Example

Create a new recordset, extract the data in blocks of 50 records

```
Const dbReadOnly = 4
Const cstNumRecs = 50
Dim orsRecords As Object, vDataBlock() As Variant
        Set orsRecords = Application.CurrentDb().OpenRecordset("EXPENSES", , , dbReadOnly)
        With orsRecords
                Do While Not .EOF()
                        Set vDataBlock = .GetRows(cstNumRecs)
                        '       ... process data block ...
                        '       vDataBlock(i, j) = value in (i + 1)th field of (j + 1)th reco
                Loop
                .mClose()
        End With
```

Bookmark this page » » GetRows