Excel OCX by SkySof Software last updated: 12/5/12

Excel OCX is a FREE powerful ActiveX Control written in Visual Basic 6.0 (ensuring maximum compatibility, ease-of-use and effectiveness) which contains a smorgasborg of functions to exchange data between Excel and Visual Basic, create professional looking reports in Excel, and much more! Using Excel OCX in your Visual Basic application will allow you to create applications in mere days which used to take weeks! All functions have been thoroughly tested and optimized to perform data transfer as quickly as possible.

Download Excel OCX from http://www.getfilez.com/excelocx.zip

To add the Excel OCX component to your Visual Basic 6.0 project:

1) Load Visual Basic 6.0.
2) From the Project Menu select "Components..."
3) Select component "Excel_OCX" and click the OK Button. The component should appear in your toolbar.
4) Double-click on the component in the toolbar to add to a Visual Basic form.

Distributing Excel OCX with your application

Excel OCX may be distributed with your application royalty free. When creating the installation program for your application you should include file MSINET.OCX (Microsoft Internet Transfer Control) if you are using the functions UploadFile() and/or DownloadFile(). The file MSINET.OCX can be found in your Window's System directory which is C:\WINDOWS\SYSTEM32 on most machines. In addition, you should also include Microsoft Data Access Components (MDAC) 2.7 or greater in your installation program if you are using any functions that utilize ADO recordsets and/or databases (RecordsetToExcel(), DatabaseTableToExcel(),etc). If the MDAC installation file MDAC_TYP.EXE can't be found on your machine you can download it free from Microsoft's web site at http://msdn.microsoft.com/data/mdac/default.aspx

Using Excel OCX in your Visual Basic Application

If you have an existing workbook then the you should use the OpenWorkbook function. Otherwise, you can use functions CreateWorkbook or CreateWorkbookFile to create a workbook.

Below is an example of how to open two workbooks, make changes to them, save the changes, close the workbooks, and finally close Excel.

```
Dim i As Integer

'Create first workbook
i = ExcelOCX1.CreateWorkbook(App.Path & "\sample1.xls")

'Create second workbook
i = ExcelOCX1.CreateWorkbook(App.Path & "\sample2.xls")

'Open first workbook
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample1.xls")

'Open second workbook
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls")

'Activate the first workbook
i = ExcelOCX1.ActivateWorkbook("sample1.xls", "Sheet3")

'Add some text to it
i = ExcelOCX1.SingleLineTextToExcel("This is some text.", 1, xlo_A)

'Done making changes to this workbook - save changes and close
ExcelOCX1.CloseWorkbook True

'Activate the second workbook
```

```
'i = ExcelOCX1.ActivateWorkbook("sample2.xls", "Sheet1")
i = ExcelOCX1.ActivateWorkbook(1, "Sheet1")

'add some text to it
i = ExcelOCX1.SingleLineTextToExcel("This is some text.", 1, xlo_A)

'Done making changes to this workbook - save changes and close
ExcelOCX1.CloseWorkbook True

'Done, close Excel
ExcelOCX1.CloseExcel False
```

Alternatively, you can also use the ActivateWorkbook() function to activate any open workbooks:

```
Dim arr() As Variant
Dim s As String
Dim i As Integer
Dim n As Integer

'Store list of all open workbooks to a one dimensional array
i = ExcelOCX1.GetWorkbookNames(arr)
If i = 1 Then
  For n = 0 To UBound(arr)
    s = s & arr(n) & vbCrLf
  Next
  MsgBox s, vbOKOnly, "List of all open workbooks"
  'Modify each workbook
  For n = 0 To UBound(arr)
    'activate workbook and worksheet
    i = ExcelOCX1.ActivateWorkbook(arr(n), "sheet1")
    'i = ExcelOCX1.ActivateWorkbook(n + 1, 1) 'can also refer to workbook and worksheet by number
    If i = 1 Then
      'add some text to the active document
      i = ExcelOCX1.SingleLineTextToExcel("This is some text.", 1, xlo_A)
      'save changes and close workbook
      ExcelOCX1.CloseWorkbook True
    End If
  Next
  'close Excel
  ExcelOCX1.CloseExcel False
ElseIf i = -1 Then
  MsgBox "Excel is not loaded.", vbOKOnly
ElseIf i = -2 Then
  MsgBox "No workbooks found.", vbOKOnly
Else
  MsgBox "An error has occurred.", vbOKOnly
End If
```

After an Excel workbook has been opened you're ready to harness the power of Excel OCX!
Below are all the functions available to Excel OCX divided into three categories:

A) Functions that move data from Visual Basic to Excel
B) Functions that move data from Excel to Visual Basic
C) Miscellaneous functions

A. Functions that move data from Visual Basic to Excel

Note: Before moving data to Excel you may want to temporarily set auto-calculation
for the workbook to manual. Doing this will greatly improve the speed at which the

data is moved (especially in workbooks with many formulas). Excel OCX has a function for doing just that named SetCalculation for setting the calculation mode (See example below). In addition to setting auto-calculation to manual, you may also want to hide Excel temporarily using the HideExcel function to improve speed. Hiding Excel minimizes screen re-drawing.

ClipboardTextToExcel() - Function to copy Clipboard text to Excel worksheet cells.

Arguments:

ByVal lngRow As Long
ByVal lngCol As Long
Optional blnMultiLineText As Boolean
Optional ByVal lngDirection As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetCalculation(xloManual)  'Turn auto-calculation off temporarily
i = ExcelOCX1.HideExcel 'Hide Excel temporarily
Clipboard.Clear
Clipboard.SetText "Hello World!!!"
i = ExcelOCX1.ClipboardTextToExcel(1, 1, True, xloDown)
i = ExcelOCX1.SetCalculation(xloAutomatic)  'Turn auto-calculation back on
i = ExcelOCX1.UnhideExcel 'Show Excel again
```

****

CollectionToExcel() - Function to copy Collection items to Excel worksheet cells.

Arguments:

cl As Collection
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a Collection
-4 = Invalid column/row
-5 = Unknown error

Example:

```
Dim i As Integer
Dim Temperatures as Collection
Set Temperatures = New Collection     'create new instance of collection object
Temperatures.Add 76, "Atlanta"        'add items to collection object. City name is the key.
Temperatures.Add 85, "Miami"
Temperatures.Add 66, "Seattle"
```

```
i = ExcelOCX1.CollectionToExcel(Temperatures, 1, 1, xloDown)
Set Temperatures = Nothing 'Destroy Collection Object
```

****

ComboBoxToExcel() - Function to copy ComboBox items to Excel worksheet cells.

Arguments:

```
cb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnSelectedOnly As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a ComboBox
-4 = Invalid column/row
-5 = Unknown error
```

Example:

```
‘This example assumes a ComboBox control is on your form
Dim i As Integer
i = ExcelOCX1.ComboBoxToExcel(Combo1, 1, 1, xloDown, False)
```

****

DatabaseTableToExcel() - Function to copy database table records to Excel worksheet cells.

Arguments:

```
strDSN As String
strSQL As String
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional strUserID As String
Optional strPassword As String
Optional blnIncludeHeadings As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unable to connect to database/Problem with DSN
-5 = Unable to open recordset
-6 = Unknown error
```

Examples:

```
Dim i As Integer
i = ExcelOCX1.DatabaseTableToExcel("My DSN", "SELECT * FROM MyTable", 1, 1, xloNormal, "", "", True)
‘example below opens a password protected database
i = ExcelOCX1.DatabaseTableToExcel("My DSN", "SELECT * FROM MyTable", 1, 1, xloTranspose, "john",
"open", True)
```

****

DataGridToExcel() - Function to copy DataGrid to Excel worksheet cells.

Arguments:

dg As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnIncludeHeadings As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

'This example assumes a populated DataGrid control is on your form
Dim i As Integer
i = ExcelOCX1.DataGridToExcel(DataGrid1, 1, 1, xloNormal, True)

Notes:

Before invoking this function make sure that the record pointer is positioned on the
first record in the Data Grid otherwise undesired results may occur.

****

DictionaryToExcel() - Function to copy Dictionary items to Excel worksheet cells.

Arguments:

dt As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnIncludeKeys As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
Dim vItem As Variant
Dim vKey As Variant
Dim Dict As Scripting.Dictionary
Set Dict = New Scripting.Dictionary   'create new instance of Dictionary Object
Dict.CompareMode = BinaryCompare      'set compare mode. Also DatabaseCompare & TextCompare
Dict.Add "Key1", "Item1"
Dict.Add "Key2", "Item2"
```

```
Dict.Add "Key3", "Item3"
Dict.Add "Key4", "Item4"
i = ExcelOCX1.DictionaryToExcel(Dict, 7, 11, xloDown, True)
Set Dict = Nothing  'Destroy Dictionary Object
```

****

DirListBoxToExcel() - Function to copy DirListBox items to Excel worksheet cells.

Arguments:

```
dlb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnSelectedOnly As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a DirListBox
-4 = Invalid column/row
-5 = Unknown error
```

Example:

```
‘This example assumes a DirListBox control is on your form
Dim i As Integer
i = ExcelOCX1.DirListBoxToExcel(Dir1, 1, 1, xloDown)
```

****

DriveListBoxToExcel() - Function to copy DriveListBox items to Excel worksheet cells.

Arguments:

```
dlb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnSelectedOnly As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a DriveListBox
-4 = Invalid column/row
-5 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.DriveListBoxToExcel(Drive1, 1, 1, xloDown)
```

****

FileListBoxToExcel() - Function to copy FileListBox items to Excel worksheet cells.

Arguments:

flb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnSelectedOnly As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a FileListBox
-4 = Invalid column/row
-5 = Unknown error

Example:

‘This example assumes a FileListBox control is on your form
Dim i As Integer
i = ExcelOCX1.FileListBoxToExcel(File1, 1, 1, xloDown, True)

****

FlexGridToExcel() - Function to copy FlexGrid to Excel worksheet cells.

Arguments:

fg As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional lngStartRow As Long
Optional lngStartCol As Long
Optional ByVal lngEndRow As Long
Optional ByVal lngEndCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

‘This example assumes a MSFlexGrid control is on your form
Dim i As Integer
i = ExcelOCX1.FlexGridToExcel(MSFlexGrid1, 1, 1, xloNormal, 1, , , 2)

****

HFlexGridToExcel() - Function to copy HFlexGrid to Excel worksheet cells.

Arguments:

fg As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long

Optional lngStartRow As Long
Optional lngStartCol As Long
Optional ByVal lngEndRow As Long
Optional ByVal lngEndCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

‘This example assumes a MSHFlexGrid control is on your form
Dim i As Integer
i = ExcelOCX1.HFlexGridToExcel(MSHFlexGrid1, 1, 1, xloNormal, 1, , , 2)

****

LabelToExcel() - Function to copy label caption to an Excel worksheet cell.

Arguments:

lbl As Object
lngRow As Long
lngCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a Label
-4 = Invalid column/row
-5 = Unknown error

Example:

‘This example assumes a Label control is on your form
Dim i As Integer
i = ExcelOCX1.LabelToExcel(Label1, 1, xlo_A) 'Note: xlo_A is the constant for column A

****

ListBoxToExcel() - Function to copy ListBox items to Excel worksheet cells.

Arguments:

lb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long
Optional blnSelectedOnly As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a ListBox

-4 = Invalid column/row
-5 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ListBoxToExcel(List1, 1, 1, xloDown, True)
```

****

ListViewToExcel() - Function to copy ListView items to Excel worksheet cells.

Arguments:

```
lv As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnSelectedOnly As Boolean
Optional blnIncludeHeadings As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error
```

Example:

```
'This example assumes a ListView control is on your form
Dim i As Integer
i = ExcelOCX1.ListViewToExcel(ListView1, 1, 1, xloNormal, False, True)
```

****

MaskedTextBoxToExcel() - Function to copy masked edit textbox text to worksheet cells.

Arguments:

```
tb As Object
lngRow As Long
lngCol As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error
```

Example:

```
'This example assumes a MaskedTextBox control is on your form
Dim i As Integer
i = ExcelOCX1.MaskedTextBoxToExcel(MaskEdBox1, 1, 1)
```

****

MultiLineTextToExcel() - Function to copy multi-line text to Excel worksheet cells.

Arguments:

ByVal strText As String
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal strDelimiter As String
Optional ByVal lngDirection As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.MultiLineTextToExcel("4.5^12/12/2001^string3", 1, 1, "^", xloDown)
```

****

OneDArrayToExcel() - Function to copy 1D array to Excel worksheet cells.

Arguments:

varArray As Variant
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnFixData As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = varArray is not an array
-5 = Unknown error

Example:

```
Dim i As Integer
Dim arr(0 To 3) As Variant
arr(0) = 5
arr(1) = 10
arr(2) = 2
arr(3) = "=SUM(A1:A3)"
i = ExcelOCX1.OneDArrayToExcel(arr, 1, 1, xloNormal)

'If the array contains date or array data types set parameter blnFixData to True
i = ExcelOCX1.OneDArrayToExcel(arr, 1, 1, xloNormal, True)
```

****

RecordsetToExcel() - Function to copy a recordset to Excel worksheet cells.

Arguments:

objRecd As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnIncludeHeadings As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
On Error Resume Next
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & _
"\author.mdb;Persist Security Info=False"
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
cn.Open strConn
Set rs = New ADODB.Recordset
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
rs.Source = "Select * From Authors"
rs.ActiveConnection = cn
rs.Open
i = ExcelOCX1.RecordsetToExcel(rs, 1, 1, xloNormal, True)
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

****

RichTextBoxToExcel() - Function to copy rich textbox text to Excel worksheet cells.

Arguments:

tb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
‘This example assumes a RichTextBox control is on your form
Dim i As Integer
```

i = ExcelOCX1.RichTextBoxToExcel(RichTextBox1, 1, 1, xloDown)

****

SingleLineTextToExcel() - Function to copy single-line text to an Excel worksheet cell.

Arguments:

strText As String
lngRow As Long
lngCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.SingleLineTextToExcel("test string", 1, 1)

****

TextBoxToExcel() - Function to copy textbox text to Excel worksheet cell(s).

Arguments:

tb As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDirection As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a TextBox
-4 = Invalid column/row
-5 = Unknown error

Example:

‘This example assumes a TextBox control is on your form
Dim i As Integer
i = ExcelOCX1.TextBoxToExcel(Text1, 1, xlo_A, xloDown)

****

TextFileToExcel() - Function to copy text file contents to Excel worksheet cells.

Arguments:

strTextFile As String
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal strDelimiter As String
Optional ByVal lngDisplay As Long

Optional blnRemoveQuotes As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Text file not found
-5 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.TextFileToExcel("c:\file.txt", 1, 1, ",", xloNormal)
```

****

TextToExcelCellComments() - Function to copy text to an Excel worksheet cell.

Arguments:

```
strText As String
lngRow As Long
lngCol As Long
Optional blnVisible As Boolean
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.TextToExcelCellComments("All you need is love!", 1, 1, True)
```

****

ThreeDArrayToExcel() - Function to copy 3D array to Excel worksheet cells.

Arguments:

```
varArray As Variant
lngRow As Long
lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnFixData As Boolean
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = varArray is not an array
-5 = Unknown error

Example:

```
Dim arr(0 To 10, 0 To 9, 0 To 2) As Variant
Dim i As Integer
Dim n As Integer
Dim j As Integer
Dim s As Integer
Screen.MousePointer = vbHourglass
j = 1
For s = LBound(arr, 3) To UBound(arr, 3)
   For i = LBound(arr, 1) To UBound(arr, 1) - 1
      For n = LBound(arr, 2) To UBound(arr, 2)
         arr(i, n, s) = j
         j = j + 1
      Next
   Next
   arr(i, 0, s) = "=SUM(A1:A10)"
   arr(i, 1, s) = "=SUM(B1:B10)"
   arr(i, 2, s) = "=SUM(C1:C10)"
   arr(i, 3, s) = "=SUM(D1:D10)"
   arr(i, 4, s) = "=SUM(E1:E10)"
   arr(i, 5, s) = "=SUM(F1:F10)"
   arr(i, 6, s) = "=SUM(G1:G10)"
   arr(i, 7, s) = "=SUM(H1:H10)"
   arr(i, 8, s) = "=SUM(I1:I10)"
   arr(i, 9, s) = "=SUM(J1:J10)"
Next
ExcelOCX1.ThreeDArrayToExcel(arr, 1, 1, xloNormal)

'If the array contains date or array data types set parameter blnFixData to True
ExcelOCX1.ThreeDArrayToExcel(arr, 1, 1, xloNormal, True)

Screen.MousePointer = vbDefault
```

****

TreeViewToExcel() – Function to copy TreeView control to Excel worksheet cells.

Arguments:

```
tv As Object
ByVal lngRow As Long
ByVal lngCol As Long
Optional blnSuppressRows As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error
```

Example:

```
'This example assumes a TreeView control is on your form
Dim Item As Node
Dim i As Integer
TreeView1.PathSeparator = "~"
Set Item = TreeView1.Nodes.Add(, , "Food", "Food")
Set Item = TreeView1.Nodes.Add("Food", tvwChild, "Fruit", "Fruit")
Set Item = TreeView1.Nodes.Add("Fruit", tvwChild, "Apple", "Apple")
Set Item = TreeView1.Nodes.Add("Fruit", tvwChild, "Banana", "Banana")
```

```
Set Item = TreeView1.Nodes.Add("Food", tvwChild, "Veggies", "Veggies")
Set Item = TreeView1.Nodes.Add("Veggies", tvwChild, "Green Beans", "Green Beans")
Set Item = TreeView1.Nodes.Add("Veggies", tvwChild, "Peas", "Peas")
TreeView1.Nodes("Food").Expanded = True
i = ExcelOCX1.TreeViewToExcel(TreeView1, 1, xlo_A, True)
```

****

TwoDArrayToExcel() - Function to copy 2D array to Excel worksheet cells.

Arguments:

```
varArray As Variant
ByVal lngRow As Long
ByVal lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnFixData As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = varArray is not an array
-5 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr(0 To 3, 0 To 3) As Variant
arr(0, 0) = 1
arr(0, 1) = 2
arr(0, 2) = 3
arr(1, 0) = 4
arr(1, 1) = 5
arr(1, 2) = 6
arr(2, 0) = 7
arr(2, 1) = 8
arr(2, 2) = 9
arr(3, 0) = "=SUM(A1:A3)"
arr(3, 1) = "=SUM(B1:B3)"
arr(3, 2) = "=SUM(C1:C3)"
i = ExcelOCX1.TwoDArrayToExcel(arr, 1, 1, xloNormal)

'If the array contains date or array data types set parameter blnFixData to True
i = ExcelOCX1.TwoDArrayToExcel(arr, 1, 1, xloNormal, True)
```

****

XMLFileToExcel() – Function to copy an XML table file's contents to worksheet cells.

Arguments:

```
strXMLFile As String
lngRow As Long
lngCol As Long
Optional ByVal lngDisplay As Long
Optional blnIncludeHeadings As Boolean
```

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unable to access XML DLL
-5 = Unable to open XML file
-6 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.XMLFileToExcel(App.Path & "\portfolio.xml", 1, xlo_A, xloNormal, True)
```

B. Functions that move data from Excel to Visual Basic

ExcelCellCommentsToText() - Function to copy a worksheet cell comment to a string variable.

Arguments:

```
strText As String
lngRow As Long
lngCol As Long
```

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
Dim strText As String
i = ExcelOCX1.ExcelCellCommentsToText(strText, 1, 1)
MsgBox "Comments for cell A1 is " & strText
```

****

ExcelCellToText() - Function to copy an Excel worksheet cell to a string variable.

Arguments:

```
strText As String
lngRow As Long
lngCol As Long
```

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
Dim strText As String
i = ExcelOCX1.ExcelCellToText(strText, 1, 1)
MsgBox "Value of cell A1 is " & strText
```

****

ExcelToClipboardText() - Function to copy worksheet cell values to Clipboard text.

Arguments:

lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional ByVal strDelimiter As String
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid start column/row
-4 = Unknown error

Example:

```
i = ExcelOCX1.ExcelToClipboardText(1, 1, 3, 3, ",", False, True)
```

****

ExcelToCollection() - Function to copy Excel worksheet cell values to a Collection Object.

Arguments:

cl As Collection
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a Collection
-4 = Invalid column/row
-5 = Unknown error

Example:

```
Dim objCol As Collection
Dim i As Integer
Dim s As String
Set objCol = New Collection
i = ExcelOCX1.ExcelToCollection(objCol, 23, 1, 30, True)
For i = 1 To objCol.Count
    s = s & objCol.Item(i) & vbCrLf
Next
Set objCol = Nothing 'Destroy Collection Object
```

MsgBox s, vbOKOnly, "Collection Items"

****

ExcelToComboBox() - Function to copy Excel worksheet cell values to a ComboBox.

Arguments:

cb As Object
ByVal lngRow As Long
ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDirection As Long
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional strSelectItem As String
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a ComboBox
-4 = Invalid column/row
-5 = Unknown error

Example:

'This example assumes a ComboBox control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToComboBox(Combo1, 1, 1, 10, xloRight, True)

****

ExcelToDatabaseTable() - Function to copy Excel worksheet cells to a database table.

Arguments:

strDSN As String
strTable As String
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional strUserID As String
Optional strPassword As String
Optional blnOverwrite As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unable to connect to database/Problem with DSN
-5 = Unable to open recordset
-6 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ExcelToDatabaseTable("AuthorsDSN", "Authors", 1, 1, 3, 3, False, True, "", "", False)
```

****

ExcelToDictionary() - Function to copy Excel worksheet cell values to a Dictionary Object.

Arguments:

```
cl As Object
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error
```

Example:

```
Dim arr() As String
Dim i As Integer
Dim objDict As Scripting.Dictionary
Dim s As String
Dim vItem As Variant
Dim vKey As Variant
Set objDict = New Scripting.Dictionary   'create new instance of objDictionary Object
objDict.CompareMode = BinaryCompare  'set compare mode. Also DatabaseCompare & TextCompare
i = ExcelOCX1.ExcelToDictionary(objDict, 23, 1, 30, True)
'Get keys
i = 0
For Each vKey In objDict.Keys
   ReDim Preserve arr(i)
   arr(i) = "Key: " & vKey
   i = i + 1
Next
i = 0
'Get items
For Each vItem In objDict.Items
   arr(i) = arr(i) & ", Item: " & vItem
   i = i + 1
Next
For i = 0 To objDict.Count - 1
   s = s & arr(i) & vbCrLf
Next
Set objDict = Nothing 'Destroy Dictionary Object
MsgBox s, vbOKOnly, "Dictionary Items"
```

****

ExcelToFlexGrid() - Function to copy Excel worksheet cells to FlexGrid control.

Arguments:

fg As Object
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

'This example assumes a FlexGrid control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToFlexGrid(MSFlexGrid1, 1, 1, 10, 5, False, True)

****

ExcelToLabel() - Function to copy an Excel worksheet cell to a label.

Arguments:

lbl As Object
lngRow As Long
lngCol As Long
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a Label
-4 = Invalid column/row
-5 = Unknown error

Example:

'This example assumes a Label control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToLabel(Label1, 1, 1)

****

ExcelToListBox() - Function to copy Excel worksheet cells to a ListBox.

Arguments:

lb As Object
ByVal lngRow As Long
ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDirection As Long
Optional blnStopAtEmptyCell As Boolean

Optional blnIgnoreEmptyCell As Boolean
Optional varSelectItems As Variant
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a ListBox
-4 = Invalid column/row
-5 = Unknown error

Example:

‘This example assumes a ListBox control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToListBox(ListBox1, 1, 1, 10, xloDown, True, False, Array("Apple"))

****

ExcelToListView() - Function to copy Excel worksheet cells to a ListView.

Arguments:

lv As Object
ByVal lngRow As Long
ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDisplay As Long
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional varSelectItems As Variant
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

‘This example assumes a ListView control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToListView(ListView1, 1, 1, 10, xloDown, True, False, Array("Apple"))

****

ExcelToMaskedTextBox() - Function to copy worksheet cell to masked edit text box.

Arguments:

tb As Object
lngRow As Long
lngCol As Long
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

‘This example assumes a MaskedEditBox control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToMaskedTextBox(MaskEdBox1, 1, 1)

****

ExcelToOneDArray() - Function to copy Excel worksheet cells to 1D array.

Arguments:

varArray As Variant
ByVal lngRow As Long
ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDirection As Long
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = varArray is not an array
-5 = Unknown error

Example:

```
Dim i As Integer
Dim arr As Variant
i = ExcelOCX1.ExcelToOneDArray(arr, 1, 1, 10, xloDown, True, False)
For i = 0 To UBound(arr)
   MsgBox arr(i)
Next
```

****

ExcelToRecordset() - Function to copy Excel worksheet cells to a recordset.

Arguments:

objConn As Object
objRecd As Object
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnOverwrite As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error

Example:

```
Dim i As Integer
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
On Error Resume Next
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & _
"\author.mdb;Persist Security Info=False"
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
cn.Open strConn
Set rs = New ADODB.Recordset
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
rs.Source = "Select * From Authors"
rs.ActiveConnection = cn
rs.Open
i = ExcelOCX1.ExcelToRecordset(cn, rs, 19, 1, 20, 3, False, True, False)
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

****

ExcelToRecordset2() - Function to copy Excel worksheet cells to an empty recordset.

Note: Excel doesn't need to be loaded to use this function.

Arguments:

```
strWorkbookFile As String
strSQL As String
objConn As Object
objRecd As Object
```

Returns: Integer

 1 = Success
-1 = Unknown error

Example:

```
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim i  As Integer
Dim s As String
On Error Resume Next
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
Set rs = New ADODB.Recordset
```

```
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
s = "SELECT * FROM [Sheet2$A18:C20]"
i = ExcelOCX1.ExcelToRecordset2(App.Path & "\sample2.xls", s, cn, rs)
s = ""
'Get headers
For i = 0 To rs.Fields.Count - 1
   s = s & rs(i).Name & vbTab
Next
s = s & vbCrLf & vbCrLf
Do While Not rs.EOF
  'Get values
  For i = 0 To rs.Fields.Count - 1
     s = s & rs(i).Value & vbTab
  Next
  s = s & vbCrLf
  rs.MoveNext
Loop
MsgBox s, vbOKOnly, "Recordset"
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

****

ExcelToRichTextBox() - Function to copy Excel worksheet cells to a rich text box.

Arguments:

```
tb As Object
ByVal lngRow As Long
ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDirection As Long
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional blnCellValue As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Unknown error
```

Example:

```
‘This example assumes a RichTextBox control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToRichTextBox(RichTextBox1, 1, 1, 10, xloDown, True, False)
```

****

ExcelToTextBox() - Function to copy Excel worksheet cells to a text box.

Arguments:

```
tb As Object
ByVal lngRow As Long
```

ByVal lngCol As Long
lngItems As Long
Optional ByVal lngDirection As Long
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Object is not a TextBox
-4 = Invalid column/row
-5 = Unknown error

Example:

'This example assumes a TextBox control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToTextBox(TextBox1, 1, 1, 10, xloDown, True, False)

****

ExcelToTextFile() - Function to copy Excel worksheet cells to a text file.

Arguments:

strTextFile As String
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional ByVal strDelimiter As String
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unable to create text file
-4 = Invalid column/row
-5 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.ExcelToTextFile("c:\file.txt", 1, 1, 10, 5, ",", False, True)

****

ExcelToThreeDArray() - Function to copy Excel worksheet cells to a 3D array.

Arguments:

varArray As Variant
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long

lngEndCol As Long
intSheets As Integer
Optional blnCellValue As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid start column/row
-4 = varArray is not an array
-5 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim j As Integer
Dim s As String
i = ExcelOCX1.ExcelToThreeDArray(arr, 22, xlo_F, 26, xlo_G, 3)
For j = LBound(arr, 3) To UBound(arr, 3)
   For i = LBound(arr, 1) To UBound(arr, 1)
      For n = LBound(arr, 2) To UBound(arr, 2)
         s = s & arr(i, n, j) & "  "
      Next
      s = s & vbCrLf
   Next
   s = s & vbCrLf
Next
MsgBox s, vbOKOnly, "3D Array Items"
```

****

ExcelToTreeView() – Function to copy Excel worksheet cells to a TreeView control.

Arguments:

tv As Object
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean
Optional blnExpandAll As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid start column/row
-4 = Unable to create array
-5 = Unknown error

Example:

```
‘‘This example assumes a TreeView control is on your form
Dim i As Integer
i = ExcelOCX1.ExcelToTreeView(TreeView1, 1, xlo_A, 16, xlo_D, False, True, False, True)
```

****

ExcelToTwoDArray() - Function to copy Excel worksheet cells to a 2D array.

Arguments:

varArray As Variant
lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid start column/row
-4 = varArray is not an array
-5 = Unknown error

Example:

```
Dim i As Integer
Dim arr As Variant
i = ExcelOCX1.ExcelToTwoDArray(arr, 1, 1, 10, 5, False, True)
```

C. Miscellaneous functions

ActivateCell() – Function to activate a specific cell in the active worksheet.

Arguments:

lngRow As Long
lngCol As Long
Optional blnSwitchToExcel As Boolean

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column

Example:

```
‘activate cell in row 5, column 5 of current worksheet and activate Excel
i = ExcelOCX1.ActivateCell( 5, xlo_E, True)
```

****

ActivateWorkbook() - Function to set an Excel workbook and worksheet as active.

Arguments:

Optional varWorkbook As Variant
Optional varWorksheet As Variant
Optional blnSwitchToExcel As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ActivateWorkbook("sample1.xls", "Sheet3", True)

‘The worksheet number can also be used
i = ExcelOCX1.ActivateWorkbook("sample1.xls", 3, True)
```

****

ActivateWorksheet() - Function to activate a specific Excel worksheet.

Arguments:

```
varWorksheet As Variant
Optional blnSwitchToExcel As Boolean
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ActivateWorksheet("Sheet2", False)

‘The worksheet number can also be used
i = ExcelOCX1.ActivateWorksheet(2, False)
```

****

ActiveWorkbookName() - Function to get active Excel workbook name.

Arguments: None

Returns: String, returns empty string if there is no active workbook.

Example:

```
MsgBox ExcelOCX1.ActiveWorkbookName
```

****

ActiveWorkbookPath() - Function to get active Excel workbook path.

Arguments: None

Returns: String, returns empty string if there is no active workbook.

Example:

MsgBox ExcelOCX1.ActiveWorkbookPath

****

ActiveWorksheetName() - Function to get active Excel worksheet name.

Arguments: None

Returns: String, returns empty string if there is no active worksheet.

Example:

MsgBox ExcelOCX1.ActiveWorksheetName

****

AddWorkbook() - Function to add an Excel workbook.

Arguments:

Optional varTemplate As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.AddWorkbook("c:\MyTemplate.xlt")

Notes:

After the workbook is added it becomes the active workbook.

****

AddWorksheet() - Function to add a worksheet to an Excel workbook.

Arguments:

Optional strWorksheet As String
Optional lngInsert As Long
Optional varWorkbook As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.AddWorksheet("My New Sheet", xloAfter)

Notes:

After the worksheet is added it becomes the active sheet.

****

AdjustColumnWidths() - Function to automatically adjust the width of columns.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Examples:

```
Dim i As Integer

'Adjust width of Column A to accomodate widest item
i = ExcelOCX1.AdjustColumnWidths("A:A")

'Adjust width of a multiple columns
i = ExcelOCX1.AdjustColumnWidths("A:D")

'Adjust width of column A to width of value in cell A1
i = ExcelOCX1.AdjustColumnWidths("A1")

'Adjust width of multiple columns to accomodate widest item in range A1:D10
i = ExcelOCX1.AdjustColumnWidths("A1:D10")
```

****

AutoCompleteCells() - Function to automatically complete cells with data.

Arguments:

strRange As String
strCharacters As String

Returns: Integer

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.AutoCompleteCells("A1:A10", "Be")
i = ExcelOCX1.AdjustColumnWidths("A:A") 'Adjust width of Column A to accomodate widest item
```

****

AutoFillCells() - Function to automatically fill cells with data.

Arguments:

strSourceRange As String
strDestinationRange As String
Optional ByVal lngFillType As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SingleLineTextToExcel("1", 1, 12)
i = ExcelOCX1.SingleLineTextToExcel("2", 2, 12)
i = ExcelOCX1.AutoFillCells("L1:L2", "L1:L20", xloDefault)
i = ExcelOCX1.AdjustColumnWidths("A:A") 'Adjust width of Column A to accomodate widest item
```

****

CalculateWorkbook() - Function to calculate an Excel workbook.

Arguments:

Optional varWorkbook As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CalculateWorkbook("MyWorkbook.xls")
```

****

CalculateWorkbooks() - Function to calculate all open Excel workbooks.

Arguments: None

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CalculateWorkbooks
```

****

CenterPage() - Function to center a printed worksheet page vertically/horizontally.

Arguments:

lngCenter As Long
blnCenter As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid center type, must be 0 or 1
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CenterPage(xloCenterVertically, True)
```

****

CheckSpelling() - Function to spell check the active worksheet.

Arguments:

None

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CheckSpelling
```

****

CollectionToOneDArray() - Function to copy Collection items to a one dimensional array.

Arguments:

cl As Collection
varArray As Variant

Returns: Integer

 1 = Success
-1 = cl is not a collection
-2 = varArray is not an array
-3 = Unknown error

Example:

```
Dim i As Integer
Dim Temperatures As Collection
dim arr() as Variant
Set Temperatures = New Collection     'create new instance of collection object
Temperatures.Add 76, "Atlanta"        'add items to collection object. City name is the key.
Temperatures.Add 85, "Miami"
Temperatures.Add 66, "Seattle"
```

```
i = ExcelOCX1.CollectionToOneDArray(Temperatures, arr)
Set Temperatures = Nothing 'Destroy Collection Object
```

****

ComboBoxToOneDArray() - Function to copy ComboBox items to a one dimensional array.

```
cb As Object
varArray As Variant
```

Returns: Integer

```
 1 = Success
-1 = cb is not a ComboBox
-2 = varArray is not an array
-3 = Unknown error
```

Example:

```
Dim i As Integer
dim arr() as Variant
i = ExcelOCX1.ComboBoxToOneDArray(Combo1, arr)
```

****

ClearCellComments() - Function to clear cell comments in the active Excel worksheet.

Arguments:

```
strRange As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.ClearCellComments("A1:E5")
```

****

ClearCellFormats() - Function to clear cell formats in the active Excel worksheet.

Arguments:

```
strRange As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.ClearCellFormats("A1:E5")
```

****

ClearCells() - Function to clear cell values in the active Excel worksheet.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ClearCells("A1:E5")
```

****

CloseExcel() - Function to close Excel.

Arguments:

blnSave As Boolean

Returns: None

Example:

```
ExcelOCX1.CloseExcel True
```

****

CloseWorkbook() - Function to close the active Excel workbook.

Arguments:

blnSave As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CloseWorkbook(True)
```

****

ComputeCells() - Function to add/subtract/multiply/divide all items in a range of cells by specified number.

Arguments:

lngStartRow As Long
lngStartCol As Long

lngEndRow As Long
lngEndCol As Long
lngCompute As Long
dblNumber As Double
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean

Returns:

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Invalid value passed for variable lngCompute (valid values are 0,1,2,3)
-5 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ComputeCells(1, xlo_M, 5, xlo_N, xloAdd, 1.5, True, True)
```

****

ComputeOneDArray() - Function to add/subtract/multiply/divide all items in a one dimensional array by specified number.

Arguments:

varArray As Variant
lngCompute As Long
dblNumber As Double

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Invalid value passed for variable lngCompute (valid values are 0,1,2,3)
-3 = Unknown error

Example:

```
Dim i As Integer
Dim arr(10) as Variant
For i = LBound(arr) To UBound(arr)
    arr(i) = i
Next
i = ExcelOCX1.ComputeOneDArray(arr, xloAdd, 2.5)    'Add 2.5 to each element in array
```

****

ComputeThreeDArray() - Function to add/subtract/multiply/divide all items in a three dimensional array by specified number.

Arguments:

varArray As Variant
lngCompute As Long
dblNumber As Double

Returns: Integer

 1 = Success

-1 = varArray is not an array
-2 = Invalid value passed for variable lngCompute (valid values are 0,1,2,3)
-3 = Unknown error

Example:

```
Dim arr(0 To 4, 0 To 4, 0 To 2) As Variant
Dim i As Integer
Dim n As Integer
Dim j As Integer
Dim k As Integer
Dim s As String
s = ""
For j = LBound(arr, 3) To UBound(arr, 3)
     k = 0
     For i = LBound(arr, 1) To UBound(arr, 1)
          For n = LBound(arr, 2) To UBound(arr, 2)
               arr(i, n, j) = k
               k = k + 1
               s = s & arr(i, n, j) & ","
          Next
      Next
      s = s & vbCrLf & vbCrLf
Next
MsgBox s, vbOKOnly, "Array values before ComputeThreeDArray()"
i = ExcelOCX1.ComputeThreeDArray(arr, xloAdd, 1.5)
s = ""
For j = LBound(arr, 3) To UBound(arr, 3)
      For i = LBound(arr, 1) To UBound(arr, 1)
           For n = LBound(arr, 2) To UBound(arr, 2)
                s = s & arr(i, n, j) & ","
           Next
      Next
      s = s & vbCrLf & vbCrLf
Next
MsgBox s, vbOKOnly, "Array values after ComputeThreeDArray()"
```

****

ComputeTwoDArray() - Function to add/subtract/multiply/divide all items in a two dimensional array by specified number.

Arguments:

varArray As Variant
lngCompute As Long
dblNumber As Double

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Invalid value passed for variable lngCompute (valid values are 0,1,2,3)
-3 = Unknown error

Example:

```
Dim arr(0 To 9, 0 To 9) As Variant
Dim i As Integer
Dim n As Integer
Dim k As Integer
Dim s As String
```

```
s = ""
k = 0
For i = LBound(arr, 1) To UBound(arr, 1)
   For n = LBound(arr, 2) To UBound(arr, 2)
      arr(i, n) = k
      k = k + 1
      s = s & arr(i, n) & ","
   Next
   s = s & vbCrLf
Next
MsgBox s, vbOKOnly, "Array values before ComputeTwoDArray()"
i = ExcelOCX1.ComputeTwoDArray(arr, xloAdd, 1.5)
s = ""
For i = LBound(arr, 1) To UBound(arr, 1)
   For n = LBound(arr, 2) To UBound(arr, 2)
      s = s & arr(i, n) & ","
   Next
   s = s & vbCrLf
Next
MsgBox s, vbOKOnly, "Array values after ComputeTwoDArray()"
```

****

CopyRange() - Function to copy a range in the active worksheet.

Arguments:

strSourceRange As String
strTargetRange As String
Optional lngPasteType As Long
Optional lngPasteSpecialOperation As Long
Optional blnSkipBlanks As Boolean
Optional blnTranspose As Boolean
Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CopyRange("A1:A10", "D1:D10", xloPasteAll, xloPasteSpecialOperationNone, False, False)
```

****

CreateChart() - Function to create a chart in Excel

Arguments:

strRange As String
Optional lngChartType As Long
Optional ByVal lngSeriesType As Long
Optional ByVal lngLocationType As Long
Optional ByVal intTop As Integer
Optional ByVal intLeft As Integer
Optional strSheet As String
Optional strTitle As String
Optional strCategoryTitle As String
Optional strValueTitle As String
Optional strExtraTitle As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
'Create a chart within sheet
i = ExcelOCX1.CreateChart("B5:C50", xlo3DColumn, xloRows, xloLocationAsObject, 6, 7, "Sheet3", "Retirement Chart", "Age", "Principal")
```

****

CreateHTML() - Function to create a HTML file from the active Excel worksheet/workbook

Arguments:

```
strHTMLFile As String
Optional blnWorkbook As Boolean
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.CreateHTML(“C:\file.htm”, False)
```

****

CreatePDF() - Function to create a PDF file from the active Excel worksheet

Arguments:

```
strPDFFile As String
Optional blnCommaDelimited As Boolean
Optional strFontName As String = "Courier"
Optional intFontSize As Integer = 10
Optional intRotation As Integer
Optional sngWidth As Single = 8.5
Optional sngHeight As Single = 11
Optional strAuthor As String
Optional strCreator As String
Optional strKeywords As String
Optional strSubject As String
Optional strTitle As String
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
'Open a workbook and activate Sheet1
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls", "Sheet1", False)
'Create a PDF file from the active worksheet
'Note: valid font names are Arial, Courier, and Times-Roman
i = ExcelOCX1.CreatePDF("C:\MyPDF.pdf", False, "Times-Roman", 10, 0, 8.5, 11, "Frank", "Frank", "Add
keywords here", "Add subject here", "Add title here")
If i = 1 Then
  If MsgBox("PDF file created. Do you want to open it?", vbYesNo, "PDF File") = vbYes Then
    'Open the PDF file
    ShellExecute 0, vbNullString, "C:\MyPDF.pdf", vbNullString, vbNullString, 1
  End If
Else
  MsgBox "Unable to create PDF file.", vbInformation
End If
ExcelOCX1.CloseExcel False
```

****

CreatePDFFromFile() - Function to create a PDF file from a text file

Arguments:

```
strTextFile As String
strPDFFile As String
Optional strFontName As String = "Courier"
Optional intFontSize As Integer = 10
Optional intRotation As Integer
Optional sngWidth As Single = 8.5
Optional sngHeight As Single = 11
Optional strAuthor As String
Optional strCreator As String
Optional strKeywords As String
Optional strSubject As String
Optional strTitle As String
```

Returns: Integer

```
 1 = Success
-1 = Text file does not exist
-2 = Unknown error
```

Example:

```
Dim i As Integer
Dim strFile As String
strFile = App.Path & "\pdf.txt"
'Open a workbook and activate Sheet1
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls", "Sheet1", False)
'Create a text file from the active worksheet
i = ExcelOCX1.ExcelToTextFile(strFile, 1, 2, 10, 4, Space(3), False, True, False)
'Create a PDF file from the text file
'Note: valid font names are Arial, Courier, and Times-Roman
i = ExcelOCX1.CreatePDFFromFile(strFile, "C:\MyPDF.pdf", "Arial", 10, 0, 8.5, 11, "Frank", "Frank", "Add
keywords here", "Add subject here", "Add title here")
If i = 1 Then
  If MsgBox("PDF file created. Do you want to open it?", vbYesNo, "PDF File") = vbYes Then
    'Open the PDF file
    ShellExecute 0, vbNullString, "C:\MyPDF.pdf", vbNullString, vbNullString, 1
  End If
```

```
Else
  MsgBox "Unable to create PDF file.", vbInformation
End If
'Delete the text file
Kill strFile
ExcelOCX1.CloseExcel False
```

****

CreatePDFUsingDistiller() - Function to create a PDF file from the active Excel worksheet range using the Distiller

Note: Adobe Acrobat Distiller must installed before using this function

Arguments:

strPDFFile As String
strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
'Open a workbook and activate Sheet3
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls", "Sheet3", False)
'Create a PDF file from the specified active worksheet range
i = ExcelOCX1.CreatePDFUsingDistiller(App.Path & "\test.pdf", "A9:C25")
If i = 1 Then
  If MsgBox("PDF file created. Do you want to open it?", vbYesNo + vbSystemModal, "PDF File") = vbYes Then
    'Open the PDF file
    ShellExecute 0, vbNullString, App.Path & "\test.pdf", vbNullString, vbNullString, 1
  End If
Else
  MsgBox "Unable to create PDF file.", vbInformation
End If
ExcelOCX1.CloseExcel False
```

****

CreatePivotTable() – Function to create a pivot table from an Excel worksheet in Excel
Arguments:

strSourceRange As String
strTargetRange As String
arrRowFields As Variant
arrColumnFields As Variant
arrPivotFields As Variant
Optional strTableName As String
Optional blnRowTotals As Boolean
Optional blnColumnTotals As Boolean
Optional blnSaveData As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active

-3 = variable is not an array
-4 = Unknown error

Example:

```
Dim i As Integer
Dim RowFields(0) As String
Dim ColFields(0) As String
Dim PivotFields(0) As String
RowFields(0) = "Office"
ColFields(0) = "Region"
PivotFields(0) = "Sales"
'Open a workbook and activate Sheet1
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls", "Sheet3", False)
'Create a pivot table from the active worksheet
'Note: For more info on pivot tables please refer to your Microsoft Excel documentation
i = ExcelOCX1.CreatePivotTable("A9:C25", "Sheet3!R1C5", RowFields, ColFields, PivotFields, "MyPivotTable")
'line below creates a pivot table in a new sheet
'i = ExcelOCX1.CreatePivotTable("A1:C17", "", RowFields, ColFields, PivotFields, "MyPivotTable")
If i = 1 Then
  MsgBox "Pivot table created.", vbOKOnly, "Pivot Table"
Else
  MsgBox CStr(i) & "Unable to create pivot table.", vbInformation, "Pivot Table"
End If
CloseExcel False
```

****

CreatePivotTableFromDB() – Function to create a pivot table from an ODBC compliant database

Arguments:

```
arrConnection As Variant
strTargetRange As String
arrRowFields As Variant
arrColumnFields As Variant
arrPivotFields As Variant
Optional strTableName As String
Optional blnRowTotals As Boolean
Optional blnColumnTotals As Boolean
Optional blnSaveData As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = variable is not an array
-4 = Unknown error
```

Example:

```
Dim i As Integer
Dim strConn As String
Dim strSQL As String
Dim arr As Variant
Dim RowFields(0) As String
Dim ColFields(0) As String
Dim PivotFields(0) As String
RowFields(0) = "Office"
ColFields(0) = "Region"
PivotFields(0) = "Sales"
```

```
'Connection string for ODBC compliant database
strConn = "ODBC;DBQ=" & App.Path & "\Author.mdb;Driver={Microsoft Access Driver (*.mdb)};"
'SQL statement - If length > 255 chars can add another array element
strSQL = "SELECT * FROM Sales;"
arr = Array(strConn, strSQL)
'Open a workbook and activate Sheet1
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls", "Sheet3", False)
'Create a pivot table from an ODBC compliant database
'Note: For more info on pivot tables please refer to your Microsoft Excel
'documentation
i = ExcelOCX1.CreatePivotTableFromDB(arr, "Sheet3!R1C5", RowFields, ColFields, PivotFields,
"MyDBPivotTable")
'line below creates a pivot table in a new sheet
'i = ExcelOCX1.CreatePivotTable("A1:C17", "", RowFields, ColFields, PivotFields, "MyPivotTable")
If i = 1 Then
  MsgBox "Pivot table created.", vbOKOnly, "Pivot Table"
Else
  MsgBox CStr(i) & "Unable to create pivot table.", vbInformation, "Pivot Table"
End If
CloseExcel False
```

****

CreateWorkbook() - Function to create an Excel workbook file.

Arguments:

strWorkbook As String
strPassword As String
strWriteResPassword As String

Returns: Integer

 1 = Success
-1 = Unknown error

Example:

```
Dim strFile As String
Dim i As Integer
strFile = App.Path & "\MyFile.xls"
i = ExcelOCX1.CreateWorkbook(strFile, “password”, “password”)
If i = 1 Then
  MsgBox "Excel workbook file '" & strFile & "' created."
Else
  MsgBox "Unable to create Excel workbook file '" & strFile & "'."
End If
```

****

CreateWorkbookFile() - Function to create an Excel 2.1 workbook file. Excel does not need to be loaded.

Arguments:

strWorkbook As String

Returns: Integer

 1 = Success
-1 = Unknown error

Example:

```
Dim strFile As String
Dim i As Integer
strFile = App.Path & "\MyFile.xls"
i = ExcelOCX1.CreateWorkbookFile(strFile)
If i = 1 Then
  MsgBox "Excel workbook file '" & strFile & "' created."
Else
  MsgBox "Unable to create Excel workbook file '" & strFile & "'."
End If
```

****

DatabaseTableToArray() - Function to move database table records to an array.

Arguments:

```
strDSN As String
strSQL As String
varArray As Variant
Optional strUserID As String
Optional strPassword As String
Optional blnIncludeHeadings As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unable to connect to database/Problem with DSN
-3 = Unable to open recordset
-4 = Unknown error
```

Example:

```
Dim i As Integer
dim arr() As Variant
i = ExcelOCX1.DatabaseTableToArray("MyDSN","SELECT * FROM MyTable",arr,"","",False)
```

****

DatabaseTableToTextFile() - Function to move database table records to a text file.

```
strDSN As String
strSQL As String
strTextFile As String
Optional strUserID As String
Optional strPassword As String
Optional blnIncludeHeadings As Boolean
Optional ByVal strDelimiter As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Unable to connect to database/Problem with DSN
-2 = Unable to open recordset
-3 = Unknown error
```

Example:

```
Dim i As Integer
'Create a comma-delimited text file from a recordset
```

```
i = ExcelOCX1.DatabaseTableToTextFile("MyDSN", "Select * From Authors", c:\Authors.csv", "", "", True,",")
'Open the comma-delimited text file in Excel
i = ExcelOCX1.OpenWorkbook("c:\Authors.csv")
```

****

DataGridToArray() - Function to copy DataGrid to one dimensional array.

Arguments:

dg As Object
varArray as Variant
Optional blnIncludeHeadings As Boolean

Returns: Integer

1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
dim arr() As Variant
i = ExcelOCX1.DataGridToArray(DataGrid1, arr, False)
```

Notes:

Before invoking this function make sure that the record pointer is positioned on the
first record in the Data Grid otherwise undesired results may occur.

****

DeleteColumnDuplicates() - Function to delete duplicate cell values for a specific column.

Arguments:

lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional intRowsToInsert As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnInsertRows As Boolean
Optional blnInsertPageBreaks As Boolean

Returns: Long

= Success, last row of data
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.DeleteColumnDuplicates(12, xlo_H, 19)
```

****

DeleteRange() - Function to delete a range in an Excel worksheet.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.DeleteRange("A1:D10")
```

****

DeleteRowDuplicates() - Function to delete duplicate cell values in a specific row.

Arguments:

lngStartCol As Long
lngRow As Long
lngEndCol As Long

Returns: Integer

 # = Success, number of deleted cell values
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.DeleteRowDuplicates(xlo_K, 12, xlo_R)
```

****

DeleteWorksheet() - Function to delete a worksheet in an Excel workbook.

Arguments:

Optional varWorksheet As Variant
Optional varWorkbook As Variant
Optional blnConfirmMessage As Boolean
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.DeleteWorksheet("Sheet2", , False, Me.Caption)

‘The worksheet number can also be used
```

```
i = ExcelOCX1.DeleteWorksheet(2, , False, Me.Caption)
```

****

DictionaryToArray() - Function to copy Dictionary items to an array.

Arguments:

```
dt As Object
varArray As Variant
Optional blnIncludeKeys As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim i As Integer
Dim vItem As Variant
Dim vKey As Variant
Dim Dict As Scripting.Dictionary
Set Dict = New Scripting.Dictionary   'create new instance of Dictionary Object
Dict.CompareMode = BinaryCompare      'set compare mode. Also DatabaseCompare & TextCompare
Dict.Add "Key1", "Item1"
Dict.Add "Key2", "Item2"
Dict.Add "Key3", "Item3"
Dict.Add "Key4", "Item4"
i = ExcelOCX1.DictionaryToArray(Dict, arr, True)
Set Dict = Nothing  'Destroy Dictionary Object
```

****

DirListBoxToOneDArray() - Function to copy DirListBox items to a one dimensional array.

Arguments: None

```
dlb As Object
varArray As Variant
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.DirListBoxToOneDArray(Dir1, arr)
```

****

DownloadFile() - Function to download a file from a remote web server.

Note: An internet connection must be established before using this function.

Arguments:

inet As Object
strRemoteHost As String
strRemoteDirectory As String
strLocalFile As String
strRemoteFile As String
Optional strUsername As String
Optional strUserPassword As String

Returns: Integer

 1 = Success
-1 = Unknown error

Example:

```
Dim i As Integer
'Note: you must modify the remote host settings before this will work
i = ExcelOCX1.DownloadFile(Inet1, "microsoft.com", "/mydirectory", "c:\file.txt", "file.txt")
If i = 1 Then
  MsgBox "File downloaded successfully!", vbOKOnly
Else
  MsgBox "Unable to download file!", vbOKOnly
End If
```

****

DriveListBoxToOneDArray() - Function to copy DriveListBox items to a one dimensional array.

Arguments: None

dlb As Object
varArray As Variant

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.DriveListBoxToOneDArray(Drive1, arr)
```

****

EmailWorkbook() - Function to email the active workbook to specified recipients.

Note: You must have Miscrosoft Mail installed.

Arguments:

strUsername As String
strPassword As String
varRecipients As Variant
Optional strSubject As String
Optional blnReturnReceipt As Boolean

Returns: Integer

 1 = Success

-1 = Excel is not active
-2 = Excel workbook not active
-3 = Microsoft Mail not installed
-4 = Unknown error

Example:

```
'If more than one recipient store them in a one dimensional array
Dim i As Integer
Dim Recipients(1) As String
Recipients(0) = "john.doe@mail.com"
Recipients(1) = "jane.doe@mail.com"
i = ExcelOCX1.EmailWorkbook("MailServerUsername", "MailServerPassword", Recipients, "My Workbook",
False)
'If you need to email to just one recipient simply pass a string:
'i = ExcelOCX1.EmailWorkbook("MailServerUsername", "MailServerPassword", "john.doe@mail.com", "My
Workbook", False)
```

****

ExcelsFreeMemory() - Function to get Excel's free memory.

Arguments: None

Returns: Long Integer

= Success - Excel's free memory
-1 = Excel is not active

Example:

```
MsgBox ExcelOCX1.ExcelsFreeMemory
```

****

ExcelsTotalMemory() - Function to get Excel's total memory.

Arguments: None

Returns: Long Integer

= Success - Excel's total memory
-1 = Excel is not active

Example:

```
MsgBox ExcelOCX1.ExcelsTotalMemory
```

****

ExcelsUsedMemory() - Function to get Excel's used memory.

Arguments: None

Returns: Long Integer

= Success - Excel's used memory
-1 = Excel is not active

Example:

```
MsgBox ExcelOCX1.ExcelsUsedMemory
```

****

FindInRange() - Function to find text in cells in a worksheet range.

Arguments:

strRange As String
strTextToFind As String
Optional blnCaseSensitive As Boolean
Optional lngCellData As Long
Optional lngCellPart As Long
Optional strCell As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error

Example:

```
Dim strSearchFor As String
Dim strStartRange As String
Dim i As Integer
Dim strCellAddress As String
strSearchFor = "FLOYD"
strStartRange = "A1:D6"
i = ExcelOCX1.FindInRange(strStartRange, strSearchFor, False, xloValues, xloPart,strCellAddress)
MsgBox "Text found in cell " & strCellAddress
AppActivate "microsoft excel"
```

****

FindInRangeAndHighlight() - Function to find and highlight text in a worksheet range.

Arguments:

strRange As String
strTextToFind As String
Optional blnCaseSensitive As Boolean
Optional lngPattern As Long
Optional lngCellData As Long
Optional lngCellPart As Long
Optional varCells As Variant

Returns: Long Integer

 # = Success, number found and highlighted
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error

Example:

```
Dim strSearchFor As String
Dim i As Long
strSearchFor = "EA"
i = ExcelOCX1.FindInRangeAndHighlight("A1:D6", strSearchFor, False, xloPatternGray16, xloValues, xloPart)
```

```
If i > 0 Then
  MsgBox "Text '" & strSearchFor & "' found. " & i & " cells highlighted."
  'Now remove pattern from cells
  i = ExcelOCX1.FindInRangeAndHighlight("A1:D6", strSearchFor, False, xloPatternNone, xloValues, xloPart)
ElseIf i = -3 Then
  MsgBox "Text '" & strSearchFor & "' not found."
Else
  MsgBox "An error has occurred."
End If
```

****

FindInRangeAndReplace() - Function to find and replace text in a worksheet range.

Arguments:

```
strRange As String
strTextToFind As String
strReplaceWithText As String
Optional blnCaseSensitive As Boolean
Optional lngCellData As Long
Optional lngCellPart As Long
Optional blnReplaceAll As Boolean
Optional lngPattern As Long
Optional varCells As Variant
```

Returns: Long Integer

```
 # = Success, number found and replaced
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error
```

Example:

```
Dim strSearchFor As String
Dim strReplaceWith As String
Dim i As Long
strSearchFor = "EA"
strReplaceWith = "XX"
i = ExcelOCX1.FindInRangeAndReplace("A1:D6", strSearchFor, strReplaceWith, False, xloValues, xloPart, False,
xloPatternGray25) 'Use xloPatternNone for no pattern
If i > 0 Then
  MsgBox "Text '" & strSearchFor & "' found. " & i & " cells data replaced."
ElseIf i = -3 Then
  MsgBox "Text '" & strSearchFor & "' not found."
Else
  MsgBox "An error has occurred."
End If
```

****

FindInWorksheet() - Function to find text in a worksheet and select the cell.

Arguments:

```
strTextToFind As String
Optional blnCaseSensitive As Boolean
Optional lngCellData As Long
Optional lngCellPart As Long
Optional strCell As String
```

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error

Example:

```
Dim strSearchFor As String
Dim strCellAddress As String
Dim i As Integer
strSearchFor = "FLOYD"
i = ExcelOCX1.FindInWorksheet(strSearchFor, False, xloValues, xloPart,strCellAddress)
MsgBox "Text found in cell " & strCellAddress
AppActivate "microsoft excel"
```

****

FindInWorksheetAndHighlight() - Function to find and highlight text in a worksheet.

Arguments:

```
strTextToFind As String
Optional blnCaseSensitive As Boolean
Optional lngPattern As Long
Optional lngCellData As Long
Optional lngCellPart As Long
Optional varCells As Variant
```

Returns: Long Integer

 # = Success, number found and highlighted
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error

Example:

```
Dim strSearchFor As String
Dim i As Long
strSearchFor = "EA"
i = ExcelOCX1.FindInWorksheetAndHighlight(strSearchFor, False, xloPatternGray16, xloValues, xloPart)
If i > 0 Then
   MsgBox "Text '" & strSearchFor & "' found. " & i & " cells highlighted."
   'Now remove pattern from cells
   i = ExcelOCX1.FindInWorksheetAndHighlight(strSearchFor, False, xloPatternNone, xloValues, xloPart)
ElseIf i = -3 Then
   MsgBox "Text '" & strSearchFor & "' not found."
Else
   MsgBox "An error has occurred."
End If
```

****

FindInWorksheetAndReplace() - Function to find and replace text in a worksheet.

```
strTextToFind As String
strReplaceWithText As String
```

Optional blnCaseSensitive As Boolean
Optional lngCellData As Long
Optional lngCellPart As Long
Optional blnReplaceAll As Boolean
Optional lngPattern As Long
Optional varCells As Variant

Returns: Long Integer

 # = Success, number found and replaced
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Nothing found
-4 = Unknown error

Example:

```
Dim strSearchFor As String
Dim strReplaceWith As String
Dim i As Long
strSearchFor = "EA"
strReplaceWith = "XX"
i = ExcelOCX1.FindInWorksheetAndReplace(strSearchFor, strReplaceWith, False, xloValues, xloPart, False,
xloPatternGray25) 'Use xloPatternNone for no pattern
If i > 0 Then
  MsgBox "Text '" & strSearchFor & "' found. " & i & " cells data replaced."
ElseIf i = -3 Then
  MsgBox "Text '" & strSearchFor & "' not found."
Else
  MsgBox "An error has occurred."
End If
```

****

FileListBoxToOneDArray() - Function to copy FileListBox items to a one dimensional array.

Arguments: None

flb As Object
varArray As Variant
Optional blnSelectedOnly As Boolean

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.FileListBoxToOneDArray(File1, arr, True)
```

****

FlexGridToArray() - Function to copy FlexGrid items to an array.

Arguments:

fg As Object
varArray As Variant

Optional lngStartRow As Long
Optional lngStartCol As Long
Optional ByVal lngEndRow As Long
Optional ByVal lngEndCol As Long

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim c As String
Screen.MousePointer = vbHourglass
i = ExcelOCX1.FlexGridToArray(MSFlexGrid1, arr, 0, 0)
Screen.MousePointer = vbDefault
c = ""
For i = 0 To UBound(arr, 1)
   For n = 0 To UBound(arr, 2)
      c = c & arr(i, n) & "     "
   Next
   c = c & vbCrLf
Next
MsgBox c, vbOKOnly, "Array Items"
```

****

FreezePanes() – Function to freeze window panes of the active worksheet.

Arguments:

lngRow As Long
lngCol As Long
Optional lngScrollRow As Long
Optional lngScrollCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Invalid scroll row/column
-5 = Unknown error

Example:

```
Dim i As Integer
‘freeze pane at row 5, column 5
i = ExcelOCX1.FreezePanes(5, xlo_E)
```

****

GetCellProperties() - Function to get the properties (array) of a specific worksheet cell.

Arguments:

strRange As String

varArray As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = varArray is not an array
-4 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim row As Integer
Dim col As Integer
Dim s As String
i = ExcelOCX1.GetCellProperties("A1", arr)
For row = 0 To UBound(arr, 1)
   For col = 0 To UBound(arr, 2)
      s = s & arr(row, col) & " | "
   Next
   s = s & vbCrLf
Next
MsgBox s, vbOKOnly, "Cell Properties"
```

****

GetCellText() - Function to get the text of a cell in a worksheet..

Arguments:

strRange As String

Returns: String

 Returns cell text if successful. Otherwise one of the following:
#ERR - Excel not active.
#ERR - No workbook.
#ERR - Unknown error.

Example:

```
Dim i As Integer
i = ExcelOCX1.GetCellText("A1"), vbOKOnly, "Cell Text"
```

****

GetCellValue() - Function to get the value of a cell in a worksheet..

Arguments:

strRange As String

Returns: Variant

 Returns cell value if successful. Otherwise one of the following:
#ERR - Excel not active.
#ERR - No workbook.
#ERR - Unknown error.

Example:

```
Dim i As Integer
i = ExcelOCX1.GetCellValue("A1"), vbOKOnly, "Cell Value"
```

****

GetExcelObject() - Function to get the active Excel application object.

Arguments:

objExcelApp As Object

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

```
Dim i As Integer
Dim objExcel As Object
'Dim objExcel As Excel.Application  'Set Reference to "Microsoft Excel Object Library" for early object binding
i = ExcelOCX1.GetExcelObject(objExcel)
```

****

GetRange() - Function to convert rows/columns to a range string.

Arguments:

lngStartRow As Long
lngStartCol As Long
Optional lngEndRow As Long
Optional lngEndCol As Long

Returns: Integer

 Returns valid range string if successful. Otherwise one of the following:
#ERR – Invalid row/column.
#ERR - Excel not active.
#ERR - No workbook.
#ERR - Unknown error.

Example:

```
Dim i As Integer
Dim strRange As String
'Use this function to transform rows/cols to range for functions that require range as a parameter
strRange = ExcelOCX1.GetRange(30, xlo_A)
i = ExcelOCX1.SingleLineTextToExcel("GetRange", 30, xlo_A)
'The SetColor function expects a range
i = ExcelOCX1.SetColor(strRange, RGB(167, 44, 56))
```

****

GetRangeObject() - Function to create an Excel range object.

Arguments:

objRange As Object
strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
Dim objRange As Object
'Dim objRange As Excel.Range  'Set Reference to "Microsoft Excel Object Library" for early object binding
i = ExcelOCX1.GetRangeObject(objRange, "A1")
```

****

GetWorkbookNames() - Function to store names of all the active workbooks in a one dimensional array.

Arguments:

varArray As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim strBook As String
n = ExcelOCX1.OpenWorkbook(App.Path & "\sample1.xls")  'open workbook
n = ExcelOCX1.GetWorkbookNames(arr)  'store names of workbooks in one dimensional array "arr"
For i = LBound(arr) To UBound(arr)  'loop through array
   strBook = arr(i)
   n = ExcelOCX1.ActivateWorkbook(strBook)  'activate the workbook
   MsgBox strBook, vbOKOnly, "Workbook Name"
Next
ExcelOCX1.CloseExcel (False)  'close Excel without saving changes
```

****

GetWorkbookObject() - Function to get the active Excel workbook object.

Arguments:

objExcelWorkbook As Object

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
Dim objWorkbook As Object
'Dim objWorkbook As Excel.Workbook  'Set Reference to "Microsoft Excel Object Library" for early object binding
i = ExcelOCX1.GetWorkbookObject(objWorkbook)
```

****

GetWorksheetNames() - Function to store names of all worksheets for the active workbook in a one dimensional array.

Arguments:

varArray As Variant

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim strSheet As String
n = ExcelOCX1.OpenWorkbook(App.Path & "\sample2.xls")  'open workbook
n = ExcelOCX1.GetWorksheetNames(arr)  'store names of worksheets in one dimensional array "arr"
For i = LBound(arr) To UBound(arr)  'loop through array
   strSheet = arr(i)
   n = ExcelOCX1.ActivateWorksheet(strSheet)  'activate the worksheet
   n = ExcelOCX1.SetPattern("A1:A5", xloPatternGray16)  'set a pattern for a range of cells
   MsgBox strSheet, vbOKOnly, "Worksheet Name"
Next
ExcelOCX1.CloseExcel (False)  'close Excel without saving changes
```

****

GetWorksheetObject() - Function to get the active Excel worksheet object.

Arguments:

objExcelWorksheet As Object

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
Dim objWorksheet As Object
'Dim objWorksheet As Excel.Worksheet  'Set Reference to "Microsoft Excel Object Library" for early object binding
i = ExcelOCX1.GetWorksheetObject(objWorksheet)
```

****

HFlexGridToArray() - Function to copy HFlexGrid items to an array.

Arguments:

fg As Object
varArray As Variant
Optional lngStartRow As Long
Optional lngStartCol As Long
Optional ByVal lngEndRow As Long
Optional ByVal lngEndCol As Long

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim c As String
Screen.MousePointer = vbHourglass
i = ExcelOCX1.HFlexGridToArray(MSHFlexGrid1, arr, 0, 0)
Screen.MousePointer = vbDefault
c = ""
For i = 0 To UBound(arr, 1)
   For n = 0 To UBound(arr, 2)
      c = c & arr(i, n) & "     "
   Next
   c = c & vbCrLf
Next
MsgBox c, vbOKOnly, "Array Items"
```

****

HideColumn() - Function to hide a column in a worksheet.

Arguments:

lngCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.HideColumn(4)
```

****

HideExcel() - Function to hide Excel.

Arguments: None

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.HideExcel
```

****

HideRow() - Function to hide a row in a worksheet.

Arguments:

lngRow As Long

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.HideRow(4)
```

****

HideWorksheet() - Function to hide a worksheet.

Arguments:

Optional varWorksheet As Variant
Optional blnVeryHidden As Boolean

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.HideWorksheet("My Worksheet", True)  'If blnVeryHidden is False then user can unhide sheet.

‘The worksheet number can also be used
i = ExcelOCX1.HideWorksheet(1, True)
```

****

HighlightDuplicateRows() - Function to highlight duplicate rows.

Arguments:

lngStartRow As Long

lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnBinaryCompare As Boolean
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional varCells As Variant
Optional blnStopAtEmptyRow As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
Dim arrHighlightedRows() As String
Screen.MousePointer = vbHourglass
i = ExcelOCX1.HighlightDuplicateRows(28, xlo_F, 40, xlo_H, True, xloPatternGray16, , , arrHighlightedRows)
Screen.MousePointer = vbDefault
Dim s As String
i =ExcelOCX1.SortOneDArray(arrHighlightedRows, xloAscending)
For i = LBound(arrHighlightedRows) To UBound(arrHighlightedRows)
   s = s & arrHighlightedRows(i) & vbCrLf
Next
MsgBox s, vbOKOnly, "Highlighted Rows"
```

****

HighlightDuplicates() - Function to highlight duplicate cell values for specified range.

Arguments:

lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long
Optional blnBinaryCompare As Boolean
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional varCells As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
Dim arrHighlightedCells() As String
Screen.MousePointer = vbHourglass
i =ExcelOCX1.HighlightDuplicates(1, xlo_H, 6, xlo_N, True, xloPatternGray16, , , arrHighlightedCells)
```

```
Screen.MousePointer = vbDefault
Dim s As String
i =ExcelOCX1.SortOneDArray(arrHighlightedCells, xloAscending)
For i = LBound(arrHighlightedCells) To UBound(arrHighlightedCells)
   s = s & arrHighlightedCells(i) & vbCrLf
Next
MsgBox s, vbOKOnly, "Highlighted Cells"
```

****

HighlightMaxInCol() - Function to highlight the maximum value in a column.

Arguments:

```
lngStartRow As Long
lngEndRow As Long
lngMaxCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional strCell As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
Dim cell As String
i = ExcelOCX1.HighlightMaxInCol(1, 10, xlo_N, xloPatternNone, vbYellow, vbRed, cell)
MsgBox cell, vbOKOnly, "Highlighted Cell"
```

****

HighlightMaxInRow() - Function to highlight the maximum value in a row.

Arguments:

```
lngMaxRow As Long
lngStartCol As Long
lngEndCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional strCell As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
```

```
Dim cell As String
i = ExcelOCX1.HighlightMaxInRow(1, xlo_M, xlo_N, xloPatternNone, vbYellow, vbRed, cell)
MsgBox cell, vbOKOnly, "Highlighted Cell"
```

****

HighlightMaxs() – Function to highlight max values in a grouped column.

Arguments:

```
lngStartRow As Long
lngGroupCol As Long
lngMaxCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional varCells As Variant
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = HighlightMaxs(2, xlo_A, xlo_F, xloPatternNone, vbYellow, vbRed)
```

****

HighlightMinInCol() - Function to highlight the minimum value in a column.

Arguments:

```
lngStartRow As Long
lngEndRow As Long
lngMinCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional strCell As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
Dim cell As String
i = ExcelOCX1.HighlightMinInCol(1, 10, xlo_N, xloPatternNone, vbYellow, vbRed, cell)
MsgBox cell, vbOKOnly, "Highlighted Cell"
```

****

HighlightMinInRow() - Function to highlight the minimum value in a row.

Arguments:

lngMinRow As Long
lngStartCol As Long
lngEndCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional strCell As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
Dim cell As String
i = ExcelOCX1.HighlightMinInRow(1, xlo_M, xlo_N, xloPatternNone, vbYellow, vbRed, cell)
MsgBox cell, vbOKOnly, "Highlighted Cell"
```

****

HighlightMins() – Function to highlight min values in a grouped column.

Arguments:

lngStartRow As Long
lngGroupCol As Long
lngMinCol As Long
Optional lngPattern As Long
Optional lngFontColor As Long
Optional lngColor As Long
Optional varCells As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.HighlightMins(2, xlo_A, xlo_F, xloPatternNone, vbYellow, vbRed)
```

****

HighlightRange() – Function to highlight a range.

Arguments:

strRange As String
Optional lngPattern As Long

Optional lngFontColor As Long
Optional lngColor As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.HighlightRange("M1:N5", xloPatternNone, vbBlue, vbGreen)

****

InsertAverages() – Function to average items in a column.

Arguments:

lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional blnSpaced As Boolean
Optional blnStopAtEmptyCell As Boolean
Optional blnIncludeTotal As Boolean
Optional strInsertOnLeft As String
Optional strTotalInsertOnLeft As String
Optional intOffset As Integer
Optional blnUseFormula

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertAverages(2, xlo_F, LastRow, True, False, True, "Average", "Grand Average", 2)

****

InsertAverages2() – Function to average items in a column. Insert average to right/left of group.

Arguments:

lngStartRow As Long
lngGroupCol As Long
lngAvgCol As Long
intOffset As Integer
Optional blnIncludeTotal As Boolean
Optional blnUseFormula

Returns: Integer

 1 = Success
-1 = Excel is not active

-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.InsertAverages2(2, xlo_A, xlo_F, 3, True)
```

****

InsertCounts() – Function to count items in a column.

Arguments:

```
lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional blnSpaced As Boolean
Optional blnStopAtEmptyCell As Boolean
Optional blnIncludeTotal As Boolean
Optional strInsertOnLeft As String
Optional strTotalInsertOnLeft As String
Optional intOffset As Integer
Optional blnUseFormula
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.InsertCounts(2, xlo_B, LastRow, True, False, True, "Count", "Total Count")
```

****

InsertCounts2() – Function to count items in a column. Insert count to right/left of group.

Arguments:

```
lngStartRow As Long
lngGroupCol As Long
lngCountCol As Long
intOffset As Integer
Optional blnIncludeTotal As Boolean
Optional blnUseFormula
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
n = ExcelOCX1.InsertCounts2(2, xlo_A, xlo_F, 3, True)
```

****

InsertColumn() - Function to insert a new column in an Excel worksheet.

Arguments:

lngCol As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column
-4 = Unknown error

Example:

```
Dim i As Integer
Dim n As Long
n = 4
i = ExcelOCX1.InsertColumn(n)
```

****

InsertHyperlink() - Function to insert a hyperlink into a worksheet cell.

Arguments:

strRange As String
strAddress As String
Optional strScreenTip As String
Optional strDisplayText As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.InsertHyperlink("A1", "http://ic.net/~kusluski", "Visit SkySof Software", "SkySof Software")
```

****

InsertHyperlinks() - Function to insert hyperlinks into a worksheet.

Arguments:

varArray As Variant
ByVal lngRow As Long
lngCol As Long

Returns: Integer

1 = Success

-1 = Excel is not active
-2 = Excel workbook not active
-3 = varArray is not an array
-4 = Unknown error

Example:

```
Dim i As Integer
Dim arr(2, 2) As String
'First Web Site
arr(0, 0) = "http://ic.net/~kusluski" 'this is required
'arr(0, 1) = "Visit SkySof Software's Home Page" 'this is optional
'arr(0, 2) = "SkySof Software" 'this is optional
'Second Web Site
arr(1, 0) = "http://www.microsoft.com"
arr(1, 1) = "Visit Microsoft's Home Page"
arr(1, 2) = "Microsoft"
'Third Web Site
arr(2, 0) = "http://www.usatoday.com"
arr(2, 1) = "Visit USA Today's Home Page"
arr(2, 2) = "USA Today"
i = ExcelOCX1.InsertHyperlinks(arr, 1, xlo_A)
```

****

InsertLinkedPicture() – Function to insert a linked picture file into the active worksheet.

Arguments:

```
strFile As String
sngLeft As Single
sngTop As Single
sngWidth As Single
sngHeight As Single
```

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
‘Note: Passed numbers for Left, Top, Width, and Height indicate pixels
i = ExcelOCX1. InsertLinkedPicture(App.Path & "\file.jpg", 100, 50, 200, 150)
```

****

InsertMaxs() - Function to insert maximum column values.

Arguments:

```
lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional blnSpaced As Boolean
Optional blnStopAtEmptyCell As Boolean
Optional blnIncludeTotal As Boolean
Optional strInsertOnLeft As String
Optional strTotalInsertOnLeft As String
```

Optional intOffset As Integer
Optional blnUseFormula

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertMaxs(2, xlo_F, LastRow, True, False, True, "Max", "Grand Max", 6)

****

InsertMaxs2() - Function to insert maximum column values. Insert max to right/left of group.

Arguments:

lngStartRow As Long
lngGroupCol As Long
lngMaxCol As Long
intOffset As Integer
Optional blnIncludeTotal As Boolean
Optional blnUseFormula

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertMaxs2(2, xlo_A, xlo_F, 4, True)

****

InsertMins() - Function to insert minimum column values.

Arguments:

lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional blnSpaced As Boolean
Optional blnStopAtEmptyCell As Boolean
Optional blnIncludeTotal As Boolean
Optional strInsertOnLeft As String
Optional strTotalInsertOnLeft As String
Optional intOffset As Integer
Optional blnUseFormula

Returns: Integer

 1 = Success

-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertMins(2, xlo_F, LastRow, True, False, True, "Min", "Grand Min", 4)

****

InsertMins2() - Function to insert minimum column values. Insert min to right/left of group.

Arguments:

lngStartRow As Long
lngGroupCol As Long
lngMinCol As Long
intOffset As Integer
Optional blnIncludeTotal As Boolean
Optional blnUseFormula

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertMins2(2, xlo_A, xlo_F, 4, True)

****

InsertPageBreak() – Function to insert a page break into the active Excel worksheet.

Arguments:

lngRow As Long
lngCol As Long
Optional lngPageBreak As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.InsertPageBreak(2, 2)

****

InsertPageBreakInWord() – Function to insert a page break into a Word document at current cursor position.

Arguments:

lngRow As Long
lngCol As Long
Optional lngPageBreak As Long

Returns: Integer

 1 = Success
-1 = Can't create Word object
-2 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.InsertPageBreakInWord
```

****

InsertPicture() – Function to insert a picture file into the active worksheet.

Arguments:

strFile As String
sngLeft As Single
sngTop As Single
sngWidth As Single
sngHeight As Single

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
'Note: Passed numbers for Left, Top, Width, and Height indicate pixels
i = ExcelOCX1. InsertPicture(App.Path & "\file.jpg", 100, 50, 200, 150)
```

****

InsertRow() - Function to insert a new row in an Excel worksheet.

Arguments:

lngRow As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row
-4 = Unknown error

Example:

```
Dim i As Integer
```

```
Dim n As Long
n = 4
i = ExcelOCX1.InsertRow(n)
```

****

InsertSums() - Function to sum items in a column.

Arguments:

```
lngStartRow As Long
lngCol As Long
lngEndRow As Long
Optional blnSpaced As Boolean
Optional blnStopAtEmptyCell As Boolean
Optional blnIncludeTotal As Boolean
Optional strInsertOnLeft As String
Optional strTotalInsertOnLeft As String
Optional intOffset As Integer
Optional blnUseFormula
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.InsertSums(2, xlo_F, 5, True, False, True, "Total", "Grand Total")
```

****

InsertSum2() - Function to sum items in a column. Insert sum to right/left of group.

Arguments:

```
lngStartRow As Long
lngGroupCol As Long
lngSumCol As Long
intOffset As Integer
Optional blnIncludeTotal As Boolean
Optional blnUseFormula
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
n = ExcelOCX1.InsertSums2(2, xlo_C, xlo_F, 1, True)
```

****

IsEmptyCell() - Function to determine if an Excel worksheet cell is empty.

Arguments:

lngRow As Long
lngCol As Long

Returns: Integer

 1 = Success - cell is empty
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column/row
-4 = Cell is not empty
-5 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.IsEmptyCell(1, 1)
If i = 1 Then
   MsgBox "Cell is empty."
Elseif i = -4 Then
   MsgBox "Cell contains data."
End If
```

****

IsEmptyRange() - Function to determine if a worksheet range of cells is empty.

Arguments:

lngStartRow As Long
lngStartCol As Long
lngEndRow As Long
lngEndCol As Long

Returns: Integer

 1 = Success - range is empty
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid start column/row
-4 = Range is not empty
-5 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.IsEmptyRange(1, 1, 10, 5)
If i = 1 Then
   MsgBox "Range is empty."
Elseif i = -4 Then
   MsgBox "Range contains data."
End If
```

****

IsExcelAvailable() - Function to determine if Excel is installed on a machine.

Arguments: None

Returns: String

Empty string = Excel is not installed
Populated String = Excel version number

Example:

```
Dim s As String
s = ExcelOCX1.IsExcelAvailable
If Len(s) Then
  MsgBox "Excel version “ & s & “ is installed on this machine.”
Else
  MsgBox "Excel is not installed on this machine.”
End If
```

****

IsExcelUp() - Function to determine if Excel is currently running.

Arguments: None

Returns: Boolean

True = Excel is running
False = Excel is not running

Example:

```
If ExcelOCX1.IsExcelUp Then
  MsgBox "Excel lives!"
Else
  MsgBox "Excel has not been loaded."
End If
```

****

ListBoxToOneDArray() - Function to copy ListBox items to a one dimensional array.

Arguments:

```
lb As Object
varArray As Variant
Optional blnSelectedOnly As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Object is not a ListBox
-2 = varArray is not an array
-3 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.ListBoxToOneDArray(List1, arr, True)
```

****

ListViewToArray() - Function to copy ListView items to an array.

Arguments:

lv As Object
varArray As Variant
Optional blnSelectedOnly As Boolean
Optional blnIncludeHeadings As Boolean

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.ListViewToArray(ListView1, arr, False, True)

****

LockRange() - Function to lock a worksheet range.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.LockRange("A1:A5")

****

MoveRange() - Function to move a range in the active worksheet.

Arguments:

strSourceRange As String
strTargetRange As String
Optional lngPasteType As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.MoveRange("A1:A10", "D1:D10", xloPasteAll)

****

MultiLineTextToOneDArray() - Function to copy multi-line text to a one dimensional array.

Arguments:

ByVal strText As String
varArray As Variant
Optional ByVal strDelimiter As String
Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.MultiLineTextToExcel("4.5^12/12/2001^string3", arr, "^")
```

****

NameWorksheet() - Function to rename a specific Excel worksheet.

Arguments:

strNewName As String
Optional strWorksheet As String
Optional blnSwitchToExcel As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Specified Excel worksheet doesn't exist
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.NameWorksheet("Sheet1", "My Sheet", True)
```

****

OneDArrayToHTMLFile() - Function to copy 1D array to an HTML file.

Arguments:

varArray As Variant
strHTMLFile As String
Optional strPageHeading As String
Optional strPageTitle As String
Optional strPageBackground As String
Optional strPageBackgroundColor As String
Optional lngTableAlign As Long
Optional strBackground As String
Optional strBackgroundColor As String
Optional intBorder As Integer
Optional strBorderColor As String
Optional intCellSpacing As Integer
Optional intCellPadding As Integer
Optional intWidthPct As Integer = 100

Optional lngTableCellAlign As Long
Optional blnWrapCellData As Boolean
Optional blnBoldFirstRow As Boolean

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim strTable As String
'store excel cell values to a one dimensional array
i = ExcelOCX1.ExcelToOneDArray(arr, 1, xlo_A, 16, xloDown, False, True, False)
'sort the array in ascending order
i = ExcelOCX1.SortOneDArray(arr, xloAscending)
'create a HTML file
i = ExcelOCX1.OneDArrayToHTMLFile(arr, App.Path & "\table.htm", CStr(UBound(arr) + 1) & " items",
"My Web Page", App.Path & "\image.jpg", "", xloTableAlignCenter, "", "cyan", 1, "blue", 2, 2, 15,
xloTableAlignLeft, False, False)
```

****

OneDArrayToHTMLTable() - Function to copy 1D array to an HTML table.

Arguments:

varArray As Variant
strTable As String
Optional lngTableAlign As Long
Optional strBackground As String
Optional strBackgroundColor As String
Optional intBorder As Integer
Optional strBorderColor As String
Optional intCellSpacing As Integer
Optional intCellPadding As Integer
Optional intWidthPct As Integer = 100
Optional lngTableCellAlign As Long
Optional blnWrapCellData As Boolean
Optional blnBoldFirstRow As Boolean

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim strTable As String
'store excel cell values to a one dimensional array
i = ExcelOCX1.ExcelToOneDArray(arr, 1, xlo_A, 16, xloDown, False, True, False)
'sort the array in ascending order
i = ExcelOCX1.SortOneDArray(arr, xloAscending)
'create a HTML table - store in variable strTable
i = ExcelOCX1.OneDArrayToHTMLTable(arr, strTable, xloTableAlignCenter, "", "cyan", 1, "blue", 2, 2,
15, xloTableAlignRight, False, False)
```

```
MsgBox strTable, vbOKOnly, "Contents of variable strTable"
'now create a web page to display the table
strTable = "<HTML><HEAD><CENTER><STRONG>Table
Example</STRONG></CENTER><BR><TITLE>Table Example</TITLE></HEAD>" & _
 "<BODY BGCOLOR='White'>" & vbCrLf & strTable & vbCrLf & "</BODY></HTML>"
MsgBox strTable, vbOKOnly, "Contents of variable strTable"
'save contents of variable strTable to a HTML file
Open App.Path & "\table.htm" For Output As #1
Print #1, strTable
Close #1
```

****

OneDArrayToText() - Function to copy a one dimensional array to a string variable.

Arguments:

```
varArray As Variant
strText As String
strEndOfRowChar As String
Optional strEndOfArrayChar As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim arr(0 To 4) As Variant
Dim s As String
Dim i As Integer
arr(0) = "The Wizard of Oz"
arr(1) = "Dirty Harry"
arr(2) = "Terminator"
arr(3) = "Star Wars"
arr(4) = "The Cable Guy"
i = ExcelOCX1.OneDArrayToText(arr, s, ", ", ".", True)
MsgBox s
```

****

OneDArrayToTextFile() - Function to copy a one dimensional array to a text file.

Arguments:

```
varArray As Variant
strTextFile As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr(3) As Integer
```

```
arr(0) = 5
arr(1) = 2
arr(2) = 9
arr(3) = 4
i = ExcelOCX1.OneDArrayToTextFile(arr, "C:\MyFile.csv", True)
```

****

OneDArrayToWord() - Function to copy a one dimensional array to a MS Word document.

Arguments:

```
varArray As Variant
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean
Optional lngTableType As Long
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Integer = 12
Optional intSpaceAfter As Integer = 10
Optional lngParagraphAlign As Long
Optional intColumnWidth As Integer
Optional blnBorders As Boolean
Optional blnShading As Boolean
Optional blnFont As Boolean
Optional blnColor As Boolean
Optional blnHeadingRows As Boolean
Optional blnLastRow As Boolean
Optional blnFirstColumn As Boolean
Optional blnLastColumn As Boolean
Optional blnAutoFit As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Unable to open Word
-2 = varArray is not an array
-3 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr(0 To 4) As Variant
arr(0) = "The Wizard of Oz"
arr(1) = "Dirty Harry"
arr(2) = "Terminator"
arr(3) = "Star Wars"
arr(4) = "2001: A Space Oddysey"
i = ExcelOCX1.TextToWord("This table was created from a one dimensional array.", App.Path & "\test.doc", False, False, True, "Arial", 12, 15, xloAlignParagraphLeft)
i = ExcelOCX1.OneDArrayToWord(arr, App.Path & "\test.doc", False, False, xloTableFormatColorful2, False, "MS Sans Serif", 12, 10, xloAlignParagraphCenter, , True, True, False, True, False, False, False, False, True)
```

****

OpenWorkbook() - Function to open an Excel workbook.

Arguments:

```
Optional strWorkbook As String
Optional varWorksheet As Variant
```

Optional blnSwitchToExcel As Boolean
Optional strPassword As String
Optional strWriteResPassword As String
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample1.xls", "Sheet2", True, "openpassword", "writepassword")

‘The worksheet number can also be used
i = ExcelOCX1.OpenWorkbook(App.Path & "\sample1.xls", 2, True, "openpassword", "writepassword")
```

****

PrintPreviewRange() - Function to print preview an Excel worksheet range.

Arguments:

Optional strRange As String
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintPreviewRange("A1:D10", True)   'Print preview range & switch to Excel
i = ExcelOCX1.PrintPreviewRange("A1:D10", False)  'Print preview range & don't switch to Excel

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintPreviewRange("A1:D10", False, Me.Caption)
```

****

PrintPreviewWorkbook() - Function to print preview an Excel workbook.

Arguments:

Optional varWorkbook As Variant
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active

-2 = Excel workbook not active
-3 = Unknown error

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintPreviewWorkbook( , True)   'Print preview workbook & switch to Excel
i = ExcelOCX1.PrintPreviewWorkbook( , False)  'Print preview workbook & don't switch to Excel

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintPreviewWorkbook( , False, Me.Caption)
```

****

PrintPreviewWorksheet() - Function to print preview an Excel worksheet.

Arguments:

```
Optional varWorksheet As Variant
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintPreviewWorksheet("Sheet2", True)   'Print preview worksheet & switch to Excel
i = ExcelOCX1.PrintPreviewWorksheet("Sheet2", False)  'Print preview worksheet & don't switch to Excel

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintPreviewWorksheet("Sheet2", False, Me.Caption)

‘The worksheet number can also be used
i = ExcelOCX1.PrintPreviewWorksheet(2, False, Me.Caption)
```

****

PrintRange() - Function to print an Excel worksheet range.

Arguments:

```
Optional strRange As String
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintRange("A1:D10", True)   'Print range & switch to Excel
i = ExcelOCX1.PrintRange("A1:D10", False)  'Print range & don't switch to Excel

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintRange("A1:D10", False, Me.Caption)
```

****

PrintWorkbook() - Function to print an Excel workbook.

Arguments:

Optional varWorkbook As Variant
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintWorkbook( , True)   'Print workbook & switch to Excel
i = ExcelOCX1.PrintWorkbook( , False)  'Print workbook & don't switch to Excel

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintWorkbook( , False, Me.Caption)
```

****

PrintWorksheet() - Function to print an Excel worksheet.

Arguments:

Optional varWorksheet As Variant
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Examples:

```
Dim i As Integer

i = ExcelOCX1.PrintWorksheet("Sheet2", True)   'Print worksheet & switch to Excel
i = ExcelOCX1.PrintWorksheet("Sheet2", False)  'Print worksheet & don't switch to Excel
```

'The caption of the form is passed here to ensure that focus is set back to the form that invoked the
'Excel OCX method.
i = ExcelOCX1.PrintWorksheet("Sheet2", False, Me.Caption)

‘The worksheet number can also be used
i = ExcelOCX1.PrintWorksheet(2, False, Me.Caption)

****

ProtectWorkbook() - Function to protect an Excel workbook.

Optional varWorkbook As Variant
Optional strPassword As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.ProtectWorkbook(,"MyPassword")

****

ProtectWorksheet() - Function to protect an Excel worksheet.

Optional varWorksheet As Variant
Optional strPassword As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.ProtectWorksheet (“MySheet”,"MyPassword")

‘The worksheet number can also be used
i = ExcelOCX1.ProtectWorksheet (1,"MyPassword")

****

RecordsetToArray() - Function to copy a recordset to an array.

Arguments:

objRecd As Object
varArray As Variant
Optional blnIncludeHeadings As Boolean

Returns: Integer

1 = Success

-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim arr() As Variant
On Error Resume Next
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & _
"\author.mdb;Persist Security Info=False"
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
cn.Open strConn
Set rs = New ADODB.Recordset
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
rs.Source = "Select * From Authors"
rs.ActiveConnection = cn
rs.Open
i = ExcelOCX1.RecordsetToArray(rs, arr, False)
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

****

RecordsetToTextFile() - Function to move a recordset to a text file.

Arguments:

```
objRecd As Object
strTextFile As String
Optional blnIncludeHeadings As Boolean
Optional ByVal strDelimiter As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = Unknown error
```

Example:

```
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim i As Integer
Dim n As Integer
Dim c As String
On Error Resume Next
Screen.MousePointer = vbHourglass
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & _
"\author.mdb;Persist Security Info=False"
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
cn.Open strConn
Set rs = New ADODB.Recordset
```

```
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
rs.Source = "Select * From Authors"
rs.ActiveConnection = cn
rs.Open
'Create comma-delimited text file from a recordset
i = ExcelOCX1.RecordsetToTextFile(rs, App.Path & "\authors.csv", True, ",", True)
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
Screen.MousePointer = vbDefault
'Open comma-delimited text file in Excel
i = ExcelOCX1.OpenWorkbook(App.Path & "\authors.csv")
```

****

RecordsetToWord() - Function to copy a recordset to a MS Word document.

Arguments:

```
objRecd As Object
Optional blnIncludeHeadings As Boolean
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean
Optional lngTableType As Long
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Integer = 12
Optional intSpaceAfter As Integer = 10
Optional lngParagraphAlign As Long
Optional intColumnWidth As Integer
Optional blnBorders As Boolean
Optional blnShading As Boolean
Optional blnFont As Boolean
Optional blnColor As Boolean
Optional blnHeadingRows As Boolean
Optional blnLastRow As Boolean
Optional blnFirstColumn As Boolean
Optional blnLastColumn As Boolean
Optional blnAutoFit As Boolean
```

Returns: Integer

```
1 = Success
-1 = Unable to open Word
-2 = Unknown error
```

Example:

```
Dim strConn As String
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim i As Integer
On Error Resume Next
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path & _
"\author.mdb;Persist Security Info=False"
Set cn = New ADODB.Connection
cn.CursorLocation = adUseClient
cn.Open strConn
```

```
Set rs = New ADODB.Recordset
rs.CursorType = adOpenStatic
rs.CursorLocation = adUseClient
rs.LockType = adLockPessimistic
rs.Source = "Select * From Authors Order By Author"
rs.ActiveConnection = cn
rs.Open
'create a Word table from an ADO recordset
i = ExcelOCX1.TextToWord("This table was created from a recordset.", App.Path & "\test.doc", False, False, True,
"Arial", 20, 15, xloAlignParagraphCenter)
i = ExcelOCX1.RecordsetToWord(rs, True, App.Path & "\test.doc", False, False, xloTableFormatColorful2, False,
"Verdana", 12, 10, xloAlignParagraphRight, 130, True, True, False, True, True, True, False, False, False)
If i <> 1 Then
   MsgBox "Unable to create document.", vbOKOnly
End If
rs.Close
cn.Close
Set rs = Nothing
Set cn = Nothing
```

****

ResetPageBreaks() - Function to reset all the page breaks in the active worksheet.

Arguments:

None

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ResetPageBreaks
```

****

RichTextBoxToOneDArray() - Function to copy rich textbox text to a one dimensional array.

Arguments:

```
tb As Object
varArray As Variant
```

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.RichTextBoxToOneDArray(Text1, arr)
```

****

RunWorkbookAutoMacro() - Function to run a specific auto macro in an Excel workbook.

Arguments:

lngMacro As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.RunWorkbookAutoMacro(xloOpen) 'Run the Auto_Open macro of the workbook
```

****

SaveWorkbook() - Function to save the active Excel workbook.

Arguments:

Optional strWorkbook As String
Optional strPassword As String
Optional strWriteResPassword As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Notes:

You must specify a workbook name when setting passwords.

Example:

```
'This example saves the workbook with the same name

Dim i As Integer
i = ExcelOCX1.SaveWorkbook
If i = 1 Then
   MsgBox "Workbook saved."
End If

'This example saves a copy of the active workbook with a different name

Dim i As Integer
i = ExcelOCX1.SaveWorkbook("c:\files\NewWorkbook.xls", "Password", "WritePassword")
If i = 1 Then
   MsgBox "Workbook saved."
End If
```

****

SetBackgroundPicture() – Function to set the background picture of the active worksheet.

Arguments:

strFile As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1. SetBackgroundPicture (App.Path & "\file.jpg")
```

****

SetBlackAndWhite() - Function to determine whether printed worksheet pages will be in black and white text.

Arguments:

blnBlackAndWhite As Boolean

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetBlackAndWhite(True)
```

****

SetBorder() - Function to set a border for a worksheet range.

Arguments:

strRange As String
ByVal lngBorderIndex As Long
Optional lngColor As Long
Optional lngLineStyle As Long
Optional lngWeight As Long

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetBorder("A1:A5", xloEdgeRight, RGB(255, 0, 0), xloContinuous, xloMedium)
```

****

SetBorders() - Function to set borders for a worksheet range.

Arguments:

strRange As String
ByVal lngBorderIndex As Long
Optional lngColor As Long
Optional lngLineStyle As Long
Optional lngWeight As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetBorders("A1:A5", RGB(255, 0, 0), xloContinuous, xloMedium)
```

****

SetCalculation() - Function to set the active Excel workbook's calculation mode.

Arguments:

ByVal lngCalculation As Long
Optional strWorkbook As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetCalculation(xloManual)
If i = 1 Then
   MsgBox "Calculation set to manual mode."
End If
```

****

SetCellFormats() - Function to set cell number formats for the active worksheet.

Arguments:

ByVal strRange As String
Optional strFormat As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetCellFormats("A1:C20", "$#,##0.00_);[Red]($#,##0.00)")
```

****

SetColor() - Function to set the background color for a worksheet range.

Arguments:

```
strRange As String
lngColor As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetColor("A1:D10", RGB(0,0,255))
```

****

SetColumnPageBreak() - Function to set a column page break.

Arguments:

```
lngCol As Long
lngPageBreak As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetColumnPageBreak(10, xloPageBreakManual)
```

****

SetColumnWidth() - Function to set the width of a column in a worksheet.

Arguments:

```
lngCol As Long
lngWidth As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
```

-3 = Invalid column
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetColumnWidth(5, 50)
```

****

SetFont() - Function to set the font name for a worksheet range.

Arguments:

```
strRange As String
strFontName As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetFont("A1:D10", "Courier New")
```

****

SetFontColor() - Function to set the font color for a worksheet range.

Arguments:

```
strRange As String
lngColor As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetFontColor("A1:D10", RGB(0,0,255))
```

****

SetFontSize() - Function to set the font size for a worksheet range.

Arguments:

```
strRange As String
ByVal intFontSize As Integer
```

Returns: Integer

```
 1 = Success
```

-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetFontSize("A1:D10", 10)
```

****

SetFontSpecial() - Function to set the special font (strikethrough,subscript or superscript) for a worksheet range.

Arguments:

strRange As String
ByVal lngFontSpecial As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetFontSpecial("A1:D10", xloSuperscript)
```

****

SetFontStyle() - Function to set the font style (Regular,Bold,Italic,Bold Italic) for a worksheet range.

Arguments:

strRange As String
ByVal lngFontStyle As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetFontStyle("A1:D10", xloBold)
```

****

SetFontUnderline() - Function to set the font underline style (None,Single,Double,Single Acctg, Double Acctng) for a worksheet range.

Arguments:

strRange As String
lngFontUnderline As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.SetFontUnderline("A1:D10", xloUnderlineStyleSingle)

****

SetFooter() - Function to set the footer of a worksheet page.

Arguments:

strFooter As String
lngFooter As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid value for lngFooter (must be 0,1, or 2)
-4 = Unknown error

Example:

Dim i As Integer
'Display date, time, page number, and custom string in footer
i = ExcelOCX1.SetFooter("&D &T &P This is a right-aligned footer.", xloFooterRight)

Special formats may be used for the footer. Below is a list of these formats:

&L = Left aligns the characters that follow.
&C = Centers the characters that follow.
&R = Right aligns the characters that follow.
&E = Turns double-underline printing on or off.
&X = Turns superscript printing on or off.
&Y = Turns subscript printing on or off.
&B = Turns bold printing on or off.
&I = Turns italic printing on or off.
&U = Turns underline printing on or off.
&S = Turns strikethrough printing on or off.
&D = Prints the current date.
&T = Prints the current time.
&F = Prints the name of the document.
&A = Prints the name of the workbook tab.
&P = Prints the page number.
&P+number = Prints the page number plus the specified number.
&P-number = Prints the page number minus the specified number.
&& = Prints a single ampersand.
& "fontname" = Prints the characters that follow in the specified font. Be sure to include the double quotation marks.
&nn = Prints the characters that follow in the specified font size. Use a two-digit number to specify a size in points.
&N = Prints the total number of pages in the document.

****

SetGridLines() - Function to set the worksheet printed pages grid lines on/off.

Arguments:

blnGridLines As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetGridLines(True)
```

****

SetHeader() - Function to set the header of a worksheet page.

Arguments:

strHeader As String
lngHeader As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid value for lngHeader (must be 0,1, or 2)
-4 = Unknown error

Example:

```
Dim i As Integer
'Display date, time, page number, and custom string in header
i = ExcelOCX1.SetHeader("&D &T &P This is a right-aligned header.", xloHeaderRight)
```

Special formats may be used for the header. Below is a list of these formats:

&L = Left aligns the characters that follow.
&C = Centers the characters that follow.
&R = Right aligns the characters that follow.
&E = Turns double-underline printing on or off.
&X = Turns superscript printing on or off.
&Y = Turns subscript printing on or off.
&B = Turns bold printing on or off.
&I = Turns italic printing on or off.
&U = Turns underline printing on or off.
&S = Turns strikethrough printing on or off.
&D = Prints the current date.
&T = Prints the current time.
&F = Prints the name of the document.
&A = Prints the name of the workbook tab.
&P = Prints the page number.
&P+number = Prints the page number plus the specified number.
&P-number = Prints the page number minus the specified number.
&& = Prints a single ampersand.
& "fontname" = Prints the characters that follow in the specified font. Be sure to include the double quotation marks.
&nn = Prints the characters that follow in the specified font size. Use a two-digit number to specify a size in points.
&N = Prints the total number of pages in the document.

****

SetHeadings() - Function to set the worksheet printed pages row/column headings on/off.

Arguments:

blnHeadings As Boolean

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetHeadings(True)
```

****

SetMargin() - Function to set the worksheet pages margin in inches.

Arguments:

lngMargin As Long
dblInches As Double

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid margin type, must be 0,1,2,3,4, or 5
-4 = Unknown Error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetMargin(xloMarginRight, 0.5)
```

****

SetOrientation() - Function to set the printed worksheet page's orientation to Portrait/Landscape mode.

Arguments:

lngOrientation As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid Orientation type, must be 0 or 1
-4 = Unknown error

Example:

```
Dim i As Integer
```

```
i = ExcelOCX1.SetOrientation(xloOrientationLandscape)  'Use landscape mode
```

****

SetPattern() - Function to set the pattern for a worksheet range.

Arguments:

strRange As String
lngPattern As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetPattern("A1:D10", xloPatternGray50)
```

****

SetPrintArea() - Function to set the print area range.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetPrintArea("A1:D10")
```

****

SetRowHeight() - Function to set the height of a row in a worksheet.

Arguments:

lngRow As Long
lngHeight As Long

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.SetRowHeight(10, 20)
```

****

SetRowPageBreak() - Function to set a row page break.

Arguments:

```
lngRow As Long
lngPageBreak As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetRowPageBreak(10, xloPageBreakManual)
```

****

SetTextHorizontal() - Function to set the horizontal text alignment for a worksheet range.

Arguments:

```
strRange As String
lngTextHorizontal As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetTextHorizontal("A1:D10", xloHAlignCenter)
```

****

SetTextVertical() - Function to set the vertical text alignment for a worksheet range.

Arguments:

```
strRange As String
lngTextVertical As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetTextVertical("A1:D10", xloVAlignCenter)
```

****

SetTitle() - Function to set the printed worksheet page's row/column titles.

Arguments:

```
lngTitle As Long
strTitleRange As String
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid title type, must be 0 or 1
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetTitle(xloTitleColumn, "A1:A5")
```

****

SetZoom() - Function to zoom a printed worksheet - 10% to 400%.

Arguments:

```
intZoomPercent As Integer
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid zoom percentage, must be 10 through 400
-4 = Unknown error
```

Example:

```
Dim i As Integer
i = ExcelOCX1.SetZoom(150)
```

****

SortOneDArray() - Function to sort a one dimensional array.

Arguments:

```
varArray As Variant
Optional ByVal lngSortOrder As Long
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim i As Integer
dim arr(0 To 3) As String
arr(0) = "fish"
arr(1) = "dog"
arr(2) = "cat"
arr(3) = "bird"
i = ExcelOCX1.SortOneDArray(arr, xloAscending)
```

****

SortRange() - Function to sort a range of Excel cells. Can sort on up to three columns.

Arguments:

```
strRange As String
varField1 As Variant
Optional lngSortOrder1 As Long
Optional varField2 As Variant
Optional lngSortOrder2 As Long
Optional varField3 As Variant
Optional lngSortOrder3 As Long
```

Returns: Integer

```
 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error
```

Example:

```
Dim i As Integer
Dim objRange1 As Object
Dim objRange2 As Object
i = ExcelOCX1.GetRangeObject(objRange1, "A1")
i = ExcelOCX1.GetRangeObject(objRange2, "B1")
i = ExcelOCX1.SortRange("A1:C20", objRange1, xloAscending, objRange2, xloAscending)
```

****

SplitWindow() – Function to split the active worksheet window at the specified row/column.

Arguments:

```
lngRow As Long
lngCol As Long
```

Returns: Integer

```
1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid row/column
-4 = Unknown error
```

Example:

```
Dim i As Integer
‘split worksheet window at row 5, column 5
```

i = ExcelOCX1.SplitWindow(5, xlo_E)

****

TextBoxToOneDArray() - Function to copy textbox text to a one dimensional array.

Arguments:

tb As Object
varArray As Variant

Returns: Integer

 1 = Success
-1 = Object is not a TextBox
-2 = varArray is not an array
-3 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.TextBoxToOneDArray(Text1, arr)
```

****

TextFileToArray() - Function to copy text file contents to an array.

Arguments:

strTextFile As String
varArray As Variant
Optional ByVal strDelimiter As String
Optional blnRemoveQuotes As Boolean

Returns: Integer

 1 = Success
-1 = Text file not found
-2 = varArray is not an array
-3 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
i = ExcelOCX1.TextFileToArray("c:\file.txt", arr, ",")
```

****

TextFileToOneDArray() - Function to copy text file contents to a one dimensional array.

Arguments:

strTextFile As String
varArray As Variant

Returns: Integer

 1 = Success
-1 = Text file not found
-2 = varArray is not an array

-3 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
i = ExcelOCX1.TextFileToOneDArray(App.Path & "\file.txt", arr)
i = ExcelOCX1.OneDArrayToExcel(arr, 1, 1, xloNormal, False)
```

****

TextToWord() - Function to copy a string to a Word document.

Arguments:

```
strText As String
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Integer = 12
Optional intSpaceAfter As Integer = 10
Optional lngParagraphAlign As Long
```

Returns: Integer

```
 1 = Success
-1 = Unable to open Word
-2 = Unknown error
```

Example:

****

TreeViewToTwoDArray() – Function to copy a two dimensional array to a TreeView object.

Arguments

```
tv As Object
varArray As Variant
Optional blnSuppressRows As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
‘This example assumes a TreeView control is on your form
Dim i As Integer
Dim n As Integer
Dim arr() As Variant
Dim s As String
i = ExcelOCX1.TreeViewToTwoDArray(TreeView1, arr, True)
For i = 0 To UBound(arr, 1)
   For n = 0 To UBound(arr, 2)
      s = s & arr(i, n) & "      "
   Next
   s = s & vbCrLf
```

```
Next
MsgBox s, vbOKOnly, "Array Items"
```

****

TwoDArrayToCollection() - Function to copy a two dimensional array to a collection object.

Arguments

varArray As Variant
cl As Collection

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = cl is not a collection object
-3 = Unknown error

Example:

```
Dim objCol As Collection
Dim i As Integer
Dim s As String
Dim arr(0 To 2, 0 To 1) As Variant
'Create collection object
Set objCol = New Collection
'Add elements to 2D array
'Items
arr(0, 0) = "Apple"
arr(1, 0) = "Orange"
arr(2, 0) = "Mango"
'Indexes
arr(0, 1) = 1
arr(1, 1) = 2
arr(2, 1) = 3
i = ExcelOCX1.TwoDArrayToCollection(arr, objCol)
If i = 0 Then Exit Sub
For i = 1 To objCol.Count
    s = s & objCol.Item(i) & vbCrLf
Next
Set objCol = Nothing 'Destroy Collection Object
MsgBox s, vbOKOnly, "Collection Items"
```

****

TwoDArrayToDictionary() - Function to copy a two dimensional array to a dictionary object.

varArray As Variant
dt As Object

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim arr() As String
Dim arr2(0 To 2, 0 To 1) As Variant
Dim i As Integer
```

```
Dim objDict As Scripting.Dictionary
Dim s As String
Dim vItem As Variant
Dim vKey As Variant
'Create dictionary object
Set objDict = New Scripting.Dictionary   'create new instance of objDictionary Object
objDict.CompareMode = BinaryCompare  'set compare mode. Also DatabaseCompare & TextCompare
'Add elements to 2D array
'Keys
arr2(0, 0) = 1
arr2(1, 0) = 2
arr2(2, 0) = 3
'Items
arr2(0, 1) = "Apple"
arr2(1, 1) = "Orange"
arr2(2, 1) = "Mango"
i = ExcelOCX1.TwoDArrayToDictionary(arr2, objDict)
If i <> 1 Then Exit Sub
'Get keys
i = 0
For Each vKey In objDict.Keys
   ReDim Preserve arr(i)
   arr(i) = "Key: " & vKey
   i = i + 1
Next
i = 0
'Get items
For Each vItem In objDict.Items
   arr(i) = arr(i) & ", Item: " & vItem
   i = i + 1
Next
For i = 0 To objDict.Count - 1
   s = s & arr(i) & vbCrLf
Next
Set objDict = Nothing 'Destroy Dictionary Object
MsgBox s, vbOKOnly, "Dictionary Items"
```

****

TwoDArrayToFlexGrid() - Function to copy a two dimensional array to a FlexGrid control.

varArray As Variant
fg As Object

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
'Note: This example assumes a FlexGrid control (named MSFlexGrid) is on your form.
Dim arr(0 To 10, 0 To 9) As Variant
Dim i As Integer
Dim n As Integer
Dim j As Integer
j = 1
For i = LBound(arr, 1) To UBound(arr, 1)
   For n = LBound(arr, 2) To UBound(arr, 2)
      arr(i, n) = j
      j = j + 1
```

```
    Next
Next
i = ExcelOCX1.TwoDArrayToFlexGrid(arr, MSFlexGrid1)
```

****

TwoDArrayToHTMLFile() - Function to copy 2D array to an HTML file.

Arguments:

```
varArray As Variant
strHTMLFile As String
Optional strPageHeading As String
Optional strPageTitle As String
Optional strPageBackground As String
Optional strPageBackgroundColor As String
Optional lngTableAlign As Long
Optional strBackground As String
Optional strBackgroundColor As String
Optional intBorder As Integer
Optional strBorderColor As String
Optional intCellSpacing As Integer
Optional intCellPadding As Integer
Optional intWidthPct As Integer = 100
Optional lngTableCellAlign As Long
Optional blnWrapCellData As Boolean
Optional blnBoldFirstRow As Boolean
Optional blnBoldFirstColumn As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim strTable As String
'store excel cell values to a two dimensional array
i = ExcelOCX1.ExcelToTwoDArray(arr, 12, xlo_C, 18, xlo_F, False, True, False)
'create a HTML file
i = ExcelOCX1.TwoDArrayToHTMLFile(arr, App.Path & "\table.htm", CStr(UBound(arr)) & " rows", "My Web
Page", App.Path & "\image.jpg", "", xloTableAlignCenter, "", "#FFD700", 1, "black", 2, 2, 30, xloTableAlignRight,
False, True, True)
```

****

TwoDArrayToHTMLTable() - Function to copy 2D array to an HTML table.

Arguments:

```
varArray As Variant
strTable As String
Optional lngTableAlign As Long
Optional strBackground As String
Optional strBackgroundColor As String
Optional intBorder As Integer
Optional strBorderColor As String
Optional intCellSpacing As Integer
Optional intCellPadding As Integer
```

```
Optional intWidthPct As Integer = 100
Optional lngTableCellAlign As Long
Optional blnWrapCellData As Boolean
Optional blnBoldFirstRow As Boolean
Optional blnBoldFirstColumn As Boolean
```

Returns: Integer

1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim strTable As String
'store excel cell values to a two dimensional array
i = ExcelOCX1.ExcelToTwoDArray(arr, 12, xlo_C, 18, xlo_F, False, True, False)
'create a HTML table - store in variable strTable
i = ExcelOCX1.TwoDArrayToHTMLTable(arr, strTable, xloTableAlignCenter, App.Path & "\image.jpg", "", 1,
"black", 2, 2, 30, xloTableAlignRight, False, True, True)
MsgBox strTable, vbOKOnly, "Contents of variable strTable"
'now create a web page to display the table
strTable = "<HTML><HEAD><CENTER><STRONG>Table
Example</STRONG></CENTER><BR><TITLE>Table Example</TITLE></HEAD>" & _
 "<BODY BGCOLOR='White'>" & vbCrLf & strTable & vbCrLf & "</BODY></HTML>"
MsgBox strTable, vbOKOnly, "Contents of variable strTable"
'save contents of variable strTable to a HTML file
Open App.Path & "\table.htm" For Output As #1
Print #1, strTable
Close #1
```

****

TwoDArrayToListView() - Function to copy a two dimensional array to a ListView control.

```
varArray As Variant
lv As Object
Optional varSelectItems As Variant
```

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
'Note: This example assumes a ListView control (named ListView1) with three columns is on your form.
Dim i As Integer
Dim arr(0 To 3, 0 To 2) As Variant
'Names
arr(0, 0) = "Billy Bob"
arr(1, 0) = "Kelly Jones"
arr(2, 0) = "Wayne Cohen"
arr(3, 0) = "Barbara Lee"
'Ages
arr(0, 1) = 24
arr(1, 1) = 17
arr(2, 1) = 35
arr(3, 1) = 53
```

```
'Sex
arr(0, 2) = "Male"
arr(1, 2) = "Female"
arr(2, 2) = "Male"
arr(3, 2) = "Female"
i = ExcelOCX1.TwoDArrayToListView(arr, ListView1, Array("Billy Bob", "Wayne Cohen"))
```

****

TwoDArrayToText() - Function to copy a two dimensional array to a string variable.

Arguments:

```
varArray As Variant
strText As String
strEndOfRowChar As String
Optional strEndOfArrayChar As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim arr(0 To 3, 0 To 2) As Variant
Dim s As String
Dim i As Integer
arr(0, 0) = "Employee"
arr(1, 0) = "John Stevens"
arr(2, 0) = "Kevin Jackson"
arr(3, 0) = "Susan Smith"
arr(0, 1) = "Age"
arr(1, 1) = 28
arr(2, 1) = 33
arr(3, 1) = 52
arr(0, 2) = "Sex"
arr(1, 2) = "M"
arr(2, 2) = "M"
arr(3, 2) = "F"

i = ExcelOCX1.TwoDArrayToText(arr, s, vbTab, vbCrLf, True)
MsgBox s
```

****

TwoDArrayToTextFile() - Function to copy a two dimensional array to a text file.

Arguments:

```
varArray As Variant
strTextFile As String
Optional ByVal strDelimiter As String
Optional blnQuotes As Boolean
```

Returns: Integer

```
 1 = Success
-1 = varArray is not an array
-2 = Unknown error
```

Example:

```
Dim arr(0 To 10, 0 To 9) As Variant
Dim i As Integer
Dim n As Integer
Dim j As Integer
Screen.MousePointer = vbHourglass
j = 1
For i = LBound(arr, 1) To UBound(arr, 1)
   For n = LBound(arr, 2) To UBound(arr, 2)
      arr(i, n) = j
      j = j + 1
   Next
Next
'Create comma-delimited text file from a two dimensional array
i = ExcelOCX1.TwoDArrayToTextFile(arr, App.Path & "\TwoD.csv", ",", True)
Screen.MousePointer = vbDefault
'Open the comma-delimited text file in Notepad
Shell "notepad.exe " & App.Path & "\TwoD.csv", vbNormalFocus
```

****

TwoDArrayToTreeView() – Function to copy a two dimensional array to a TreeView control.

Arguments:

```
varArray As Variant
tv As Object
Optional blnExpandAll As Boolean
```

Returns: Integer

1 = Success
-1 = varArray is not an array
-2 = Unknown error

Example:

```
‘This example assumes a TreeView control is on your form
Dim arr(15, 3) As String
Dim i As Integer
arr(0, 0) = "Food"
arr(1, 1) = "Fruit"
arr(2, 2) = "Apples"
arr(3, 3) = "Granny Smith"
arr(4, 3) = "Macintosh"
arr(5, 2) = "Oranges"
arr(6, 3) = "Navel"
arr(7, 3) = "Valencia"
arr(8, 1) = "Veggies"
arr(9, 2) = "Beans"
arr(10, 3) = "Green"
arr(11, 3) = "Lima"
arr(12, 2) = "Lettuce"
arr(13, 3) = "Romaine"
arr(14, 3) = "Iceburg"
arr(15, 3) = "Green Leaf"
i = ExcelOCX1.TwoDArrayToTreeView(arr, TreeView1, True)
```

****

TwoDArrayToWord() - Function to copy a two dimensional array to a MS Word document.

Arguments:

varArray As Variant
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean
Optional lngTableType As Long
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Integer = 12
Optional intSpaceAfter As Integer = 10
Optional lngParagraphAlign As Long
Optional intColumnWidth As Integer
Optional blnBorders As Boolean
Optional blnShading As Boolean
Optional blnFont As Boolean
Optional blnColor As Boolean
Optional blnHeadingRows As Boolean
Optional blnLastRow As Boolean
Optional blnFirstColumn As Boolean
Optional blnLastColumn As Boolean
Optional blnAutoFit As Boolean

Returns: Integer

 1 = Success
-1 = Unable to open Word
-2 = varArray is not an array
-3 = Unknown error

Example:

```
Dim i As Integer
Dim arr(0 To 3, 0 To 1) As Variant
arr(0, 0) = "ID"
arr(1, 0) = 1
arr(2, 0) = 2
arr(3, 0) = 3
arr(0, 1) = "Fruit"
arr(1, 1) = "Apple"
arr(2, 1) = "Orange"
arr(3, 1) = "Mango"
i = ExcelOCX1.TextToWord("This table was created from a two dimensional array.", App.Path & "\test.doc", False, False, True, "Arial", 12, 15, xloAlignParagraphLeft)
i = ExcelOCX1.TwoDArrayToWord(arr, App.Path & "\test.doc", False, False, xloTableFormatProfessional, False, "Courier", 8, 10, xloAlignParagraphCenter, , True, True, False, True, True, True, False, False, True)
If i <> 1 Then
   MsgBox "Unable to create document.", vbOKOnly
Else
   MsgBox "Document created.", vbOKOnly
End If
```

****

UnfreezePanes() – Function to unfreeze panes of the active worksheet.

Arguments:

None

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnfreezePanes

****

UnhideColumn() - Function to unhide a column in a worksheet.

Arguments:

lngCol As Long

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid column
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnhideColumn(4)

****

UnhideExcel() - Function to unhide Excel.

Arguments: None

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnhideExcel

****

UnhideRow() - Function to unhide a row in a worksheet.

Arguments:

lngRow As Long

Returns: Integer

1 = Success
-1 = Excel is not active

-2 = Excel workbook not active
-3 = Invalid row
-4 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnhideRow(4)

****

UnhideWorksheet() - Function to unhide a worksheet.

Arguments:

Optional varWorksheet As Variant

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnhideWorksheet("My Worksheet")

'The worksheet number can also be used
i = ExcelOCX1.UnhideWorksheet(1)

****

UnlockRange() - Function to unlock a worksheet range.

Arguments:

strRange As String

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

Dim i As Integer
i = ExcelOCX1.UnlockRange("A1:A5")

****

UnprotectWorkbook() - Function to unprotect an Excel workbook.

Arguments:

Optional varWorkbook As Variant
Optional strPassword As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.UnprotectWorkbook( , "MyPassword")
```

****

UnprotectWorksheet() - Function to unprotect an Excel worksheet.

Arguments:

Optional varWorksheet As Variant
Optional strPassword As String

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.UnprotectWorksheet(“MySheet”, "MyPassword")

‘The worksheet number can also be used
i = ExcelOCX1.UnprotectWorksheet(1, "MyPassword")
```

****

UnsplitWindow() – Function to unsplit the active worksheet window.

Arguments: None

Returns: Integer

1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.UnsplitWindow
```

****

UploadFile() - Function to copy a local file to a remote web server.

Note: An internet connection must be established before using this function.

Arguments:

inet As Object

strRemoteHost As String
strRemoteDirectory As String
strLocalFile As String
strRemoteFile As String
Optional strUsername As String
Optional strUserPassword As String

Returns: Integer

1 = Success
-1 = Unknown error

Example:

```
Dim i As Integer
Dim arr() As Variant
Dim strTable As String
'store excel cell values to a one dimensional array
i = ExcelOCX1.ExcelToOneDArray(arr, 1, xlo_A, 16, xloDown, False, True, False)
'sort the array in ascending order
i = ExcelOCX1.SortOneDArray(arr, xloAscending)
'create a HTML file
i = ExcelOCX1.OneDArrayToHTMLFile(arr, App.Path & "\table.htm", CStr(UBound(arr) + 1) & " items", "My
Web Page", App.Path & "\image.jpg", "", xloTableAlignCenter, "", "cyan", 1, "blue", 2, 2, 15, xloTableAlignLeft,
False, False)
If MsgBox("HTML file created. Do you want to upload it to a web server?", vbQuestion + vbYesNo, "HTML
Table") = vbYes Then
  'Note: you must modify the remote host settings before this will work
  i = ExcelOCX1.UploadFile(Inet1, "www.microsoft.com", "/mydirectory", App.Path & "\table.htm", "table.htm",
"username", "password")
  If i = 1 Then
    MsgBox "File uploaded successfully!", vbOKOnly
  Else
    MsgBox "Unable to upload file!", vbOKOnly
  End If
End If
```

****

WorksheetCount() – Function that returns the number of worksheets for the active workbook.

Arguments:

None

Returns: Long Integer

= Success - number of worksheets in the active workbook
-1 = Excel is not active
-2 = Excel workbook not active

Example:

```
Dim i As Integer
Dim n As Integer
For i = 1 To ExcelOCX1.WorksheetCount
   n = ExcelOCX1.ActivateWorksheet(i)  'activate the worksheet
   n = ExcelOCX1.SetPattern("A1:A5", xloPatternGray16)  'set a pattern for a range of cells
   MsgBox "Sheet" & i, vbOKOnly, "Worksheet Name"
Next
```

****

XMLFileToArray () – Function to copy an XML table file's contents to a two dimensional array.

Arguments:

strXMLFile As String
varArray As Variant
Optional blnIncludeHeadings As Boolean

Returns: Integer

 1 = Success
-1 = varArray is not an array
-2 = Unable to access XML DLL
-3 = Unable to open XML file
-4 = Unknown error

Example:

```
Dim arr() As Variant
Dim i As Integer
Dim n As Integer
Dim s As String
i = ExcelOCX1.XMLFileToArray(App.Path & "\portfolio.xml", arr, True)
For i = 0 To UBound(arr, 1)
   For n = 0 To UBound(arr, 2)
      s = s & arr(i, n) & ","
   Next
   s = s & vbCrLf
Next
MsgBox s, vbOKOnly, "Array Items"
```

****

XMLFileToRecordset () – Function to copy an XML table file's contents to a recordset.

Arguments:

strXMLFile As String
objRecd As Object

Returns: Integer

 1 = Success
-1 = Unable to access XML DLL
-2 = Unable to open XML file/Unknown error

Example:

```
Dim i As Integer
Dim rs As ADODB.Recordset
Dim fld As ADODB.Field
Dim s As String
Set rs = New ADODB.Recordset
i = ExcelOCX1.XMLFileToRecordset(App.Path & "\portfolio.xml", rs)
For Each fld In rs.Fields
   If fld.Name <> "$Text" Then
      s = s & fld.Name & ","
   End If
Next
s = s & vbCrLf
Do While Not rs.EOF
```

```
  For Each fld In rs.Fields
    If fld.Name <> "$Text" Then
      s = s & fld.Value & ","
    End If
  Next
  s = s & vbCrLf
  rs.MoveNext
Loop
MsgBox s, vbOKOnly, "Recordset Items"
```

****

XMLFileToTreeView() - Function to copy items from a XML file to a TreeView control.

Arguments:

strXMLFile As String
tv As Object
Optional blnExpandAll As Boolean

Returns: Integer

 1 = Success
-1 = unable to create the XML ActiveX object
-2 = unable to load XML file
-3 = Unknown error

Example:

```
'This example assumes a TreeView control is on your form
Dim i As Integer
TreeView1.Nodes.Clear
i = ExcelOCX1.XMLFileToTreeView(App.Path & "\file.xml", TreeView1, False)
'To use an XML file located on a web server use the URL
'i = ExcelOCX1.XMLFileToTreeView("http://www.mypage.com/file.xml", TreeView1, True)
```

****

ZoomWorksheet() - Function to zoom in/out of a worksheet - 10% to 400%.

Arguments:

intZoomPercent As Integer

Returns: Integer

 1 = Success
-1 = Excel is not active
-2 = Excel workbook not active
-3 = Invalid zoom percentage
-4 = Unknown error

Example:

```
Dim i As Integer
i = ExcelOCX1.ZoomWorksheet(200)
```