Excel Class .NET by SkySof Software last updated: 11/1/04

Excel Class .NET is a powerful .NET Class Module written in Visual Basic .NET (ensuring maximum compatibility and effectiveness) which contains a smorgasborg of functions to exchange data between Excel and .NET, create professional looking reports in Excel, and much much more! Using Excel Class .NET in your .NET application will allow you to create applications in mere days which used to take weeks! The amount of time and money saved more than makes up for the cost of the product. Our many satisfied customers tell us that Excel Class .NET pays for itself the first time it is used! All functions have been thoroughly tested and optimized to perform data transfer as quickly as possible. If you discover any problems or have an idea for a new function please email a detailed description to SkySof Software at kusluski@mail.ic.net

The cost of Excel Class .NET is only $49.95 per developer. If you want to customize the Excel Class .NET functions to your liking the complete Visual Basic .NET source code can be purchased for only $99.95. All upgrades are free and Excel Class .NET may be distributed with your application royalty free.

To add the Excel Class .NET component to your .NET project:

1) Load your .NET project
2) From the Project Menu select "Add Reference..."
3) Click the Browse Button and select file C:\Program Files\SkySof Software\Excel Class .NET\Excelcls.dll then click the Open Button.
4) Click the OK Button to add the reference to your project. In the Solution Explorer in the References Section you should now see "Excelcls". Note that the assembly is strongly named meaning it can be installed on other machines as a unique DLL – no need to worry about DLL name conflicts.

Distributing Excel Class .NET with your application

Excel Class .NET may be distributed with your application royalty free. When creating the installation program for your application you should 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 Class .NET in your .NET Application

If you have an existing workbook then 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.

First, add a new module to your project (Project Menu > Add Module…) and enter following code between Module…End Module:

```
Public oExcel As New SkySof.Excel()
```

Next, add a form to your project (Project Menu > Add Windows Form…) and enter the following code to the top of the form before the Public Class statement:

```
Imports SkySof.Excel
```

Finally, in the load subroutine of the form add the following code:

```
Dim i As Short
Dim AppPath As String = "C:\temp" 'You may need to change the path

'Create first workbook
i = oExcel.CreateWorkbook(AppPath & "\sample1.xls")

'Create second workbook
```

```
i = oExcel.CreateWorkbook(AppPath & "\sample2.xls")

'Open first workbook
i = oExcel.OpenWorkbook(Me, Me, AppPath & "\sample1.xls")

'Open second workbook
i = oExcel.OpenWorkbook(Me, Me, AppPath & "\sample2.xls")

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

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

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

'Activate the second workbook
'i = oExcel.ActivateWorkbook("sample2.xls", "Sheet1")
i = oExcel.ActivateWorkbook(1, "Sheet1")

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

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

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

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

```
Dim arr As Object
Dim s As String
Dim i As Short
Dim n As Short

'Store list of all open workbooks to a one dimensional array
i = oExcel.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 = oExcel.ActivateWorkbook(arr(n), "sheet1")
    'i = oExcel.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 = oExcel.SingleLineTextToExcel("This is some text.", 1, ColType.xlo_A)
      'save changes and close workbook
      oExcel.CloseWorkbook True
    End If
  Next
  'close Excel
  oExcel.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 Class .NET! Below are all the functions available to Excel Class .NET divided into three categories:

A) Functions that move data from .NET to Excel
B) Functions that move data from Excel to .NET
C) Miscellaneous functions

A. Functions that move data from .NET 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 Class .NET 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 intRow As Integer
ByVal intCol As Integer
Optional blnMultiLineText As Boolean
Optional ByVal intDirection As Integer
```

Returns: Short

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

Example:

```
Dim i As Short
Dim oDataObject As New DataObject()
oDataObject.SetData(DataFormats.Text, TextBox1.Text)
Clipboard.SetDataObject(oDataObject)
i = oExcel.ClipboardTextToExcel(1, 1, True, DirectionType.xloDown)
i = oExcel.AdjustColumnWidths("A:A")
oDataObject.SetData(DataFormats.Text, "")
Clipboard.SetDataObject(oDataObject)
```

****

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

Arguments:

```
cl As Collection
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
```

Returns: Short

```
 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 Short
Dim Temperatures As Collection
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 = oExcel.CollectionToExcel(Temperatures, 7, 11, DirectionType.xloDown)
Temperatures = Nothing     'Destroy Collection Object
```

****

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

Arguments:

```
cb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnSelectedOnly As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.ComboBoxToExcel(Combo1, 1, 1,  DirectionType.xloDown, False)
```

****

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

Arguments:

```
strDSN As String
strSQL As String
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional strUserID As String
Optional strPassword As String
Optional blnIncludeHeadings As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.ADODatabaseTableToExcel("My DSN", "SELECT * FROM MyTable", 1, 1, DisplayType.xloNormal,
"", "", True)
‘example below opens a password protected database
i = OExcel.ADODatabaseTableToExcel("My DSN", "SELECT * FROM MyTable", 1, 1,
DisplayType.xloTranspose, "john", "open", True)
```

****

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

Arguments:

```
dg As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnIncludeHeadings As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.DataGridToExcel(DataGrid1, 1, 1, DisplayType.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 intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnIncludeKeys As Boolean
```

Returns: Short

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

-4 = Unknown error

Example:

```
Dim i As Short
Dim objDict As New MyDictionary()
objDict.Add("Key1", "Item1")
objDict.Add("Key2", "Item2")
objDict.Add("Key3", "Item3")
objDict.Add("Key4", "Item4")
i = oExcel.DictionaryToExcel(objDict, 7, 11, DisplayType.xloNormal, True)
objDict = Nothing      'Destroy objDictionary Object
```

****

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

Arguments:

```
dlb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnSelectedOnly As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.DirListBoxToExcel(Dir1, 1, 1,  DirectionType.xloDown)
```

****

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

Arguments:

```
dlb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnSelectedOnly As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.DriveListBoxToExcel(Drive1, 1, 1,  DirectionType.xloDown)
```

****

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

Arguments:

```
flb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnSelectedOnly As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.FileListBoxToExcel(File1, 1, 1,  DirectionType.xloDown, True)
```

****

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

Arguments:

```
fg As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional intStartRow As Integer
Optional intStartCol As Integer
Optional ByVal intEndRow As Integer
Optional ByVal intEndCol As Integer
```

Returns: Short

```
 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 Short
i = OExcel.FlexGridToExcel(MSFlexGrid1, 1, 1, DisplayType.xloNormal, 1, , , 2)
```

****

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

Arguments:

lbl As Object
intRow As Integer
intCol As Integer

Returns: Short

 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 Short
i = OExcel.LabelToExcel(Label1, 1, ColType.xlo_A) 'Note: ColType.xlo_A is the constant for column A

****

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

Arguments:

lb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
Optional blnSelectedOnly As Boolean

Returns: Short

 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 Short
i = OExcel.ListBoxToExcel(List1, 1, 1, DirectionType.xloDown, True)

****

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

Arguments:

lv As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnSelectedOnly As Boolean
Optional blnIncludeHeadings As Boolean

Returns: Short

 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 Short
i = OExcel.ListViewToExcel(ListView1, 1, 1, DisplayType.xloNormal, False, True)

****

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

Arguments:

ByVal strText As String
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal strDelimiter As String
Optional ByVal intDirection As Integer

Returns: Short

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

Example:

Dim i As Short
i = OExcel.MultiLineTextToExcel("4.5^12/12/2001^string3", 1, 1, "^", DirectionType.xloDown)

****

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

Arguments:

objArray As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnFixData As Boolean

Returns: Short

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

Example:

Dim i As Short
Dim arr(0 To 3) As Object
arr(0) = 5
arr(1) = 10

```
arr(2) = 2
arr(3) = "=SUM(A1:A3)"
i = OExcel.OneDArrayToExcel(arr, 1, 1, DisplayType.xloNormal)

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

****

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

Arguments:

```
objRecd As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnIncludeHeadings As Boolean
```

Returns: Short

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

Example:

```
Dim i As Short
Dim strConn As String
Dim cn As New ADODB.Connection()
Dim rs As ADODB.Recordset
strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & AppPath & _
"\author.mdb;Persist Security Info=False"
cn.Open(strConn)
cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
rs = cn.Execute("Select AU_ID As [ID Number], [Year Born] As [Birth Date], Author As [Author Name]
From Authors")
i = oExcel.ADORecordsetToExcel(rs, 1, 1, DisplayType.xloNormal, True)
i = oExcel.AdjustColumnWidths("A:C") 'Adjust width of Columns to accomodate widest item
rs.Close()
cn.Close()
rs = Nothing
cn = Nothing
GC.Collect()
```

****

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

Arguments:

```
strText As String
intRow As Integer
intCol As Integer
```

Returns: Short

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

-4 = Unknown error

Example:

```
Dim i As Short
i = OExcel.SingleLineTextToExcel("test string", 1, 1)
```

****

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

Arguments:

```
tb As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDirection As Integer
```

Returns: Short

```
 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 Short
i = OExcel.TextBoxToExcel(Text1, 1, ColType.xlo_A,  DirectionType.xloDown)
```

****

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

Arguments:

```
strTextFile As String
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal strDelimiter As String
Optional ByVal intDisplay As Integer
Optional blnRemoveQuotes As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.TextFileToExcel("c:\file.txt", 1, 1, ",", DisplayType.xloNormal)
```

****

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

Arguments:

strText As String
intRow As Integer
intCol As Integer
Optional blnVisible As Boolean

Returns: Short

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

Example:

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

****

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

Arguments:

objArray As Object
intRow As Integer
intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnFixData As Boolean

Returns: Short

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

Example:

```
    Dim i As Short
     Dim arr(10, 9, 2) As Object
     Dim n As Short
     Dim j As Short
     Dim s As Short
     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
    i = oExcel.ThreeDArrayToExcel(arr, 1, 1, DisplayType.xloNormal)
```

****

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

Arguments:

tv As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional blnSuppressRows As Boolean

Returns: Short

 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 i As Short
i = oExcel.TreeViewToExcel(TreeView1, 1, 1, True)
```

****

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

Arguments:

objArray As Object
ByVal intRow As Integer
ByVal intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnFixData As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim arr(0 To 3, 0 To 3) As Object
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 = OExcel.TwoDArrayToExcel(arr, 1, 1, DisplayType.xloNormal)

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

****

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

Arguments:

```
strXMLFile As String
intRow As Integer
intCol As Integer
Optional ByVal intDisplay As Integer
Optional blnIncludeHeadings As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.XMLFileToExcel(AppPath & "\portfolio.xml", 1, ColType.xlo_A, DisplayType.xloNormal, True)
```

B. Functions that move data from Excel to .NET

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

Arguments:

```
strText As String
intRow As Integer
intCol As Integer
```

Returns: Short

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

Example:

```
Dim i As Short
Dim strText As String
i = OExcel.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
intRow As Integer
intCol As Integer

Returns: Short

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

Example:

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

****

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

Arguments:

intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional ByVal strDelimiter As String
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Short

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

Example:

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

****

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

Arguments:

cl As Collection
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
Optional blnStopAtEmptyRow As Boolean

Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Short

 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 Short
Dim s As String
Set objCol = New Collection
i = OExcel.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 intRow As Integer
ByVal intCol As Integer
intItems As Integer
Optional ByVal intDirection As Integer
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional strSelectItem As String
Optional blnCellValue As Boolean

Returns: Short

 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 Short
i = OExcel.ExcelToComboBox(Combo1, 1, 1, 10, DirectionType.xloRight, True)
```

****

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

Arguments:

strDSN As String

strTable As String
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
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: Short

 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 Short
i = OExcel.ExcelToADODatabaseTable("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
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Short

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

Example:

```
Dim arr as Object
Dim i As Short
Dim objDict As Scripting.Dictionary
Dim s As String
Dim vItem As Object
Dim vKey As Object
Set objDict = New Scripting.Dictionary   'create new instance of objDictionary Object
objDict.CompareMode = BinaryCompare  'set compare mode. Also DatabaseCompare & TextCompare
i = OExcel.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
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.ExcelToFlexGrid(MSFlexGrid1, 1, 1, 10, 5, False, True)
```

****

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

Arguments:

```
lbl As Object
intRow As Integer
intCol As Integer
Optional blnCellValue As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.ExcelToLabel(Label1, 1, 1)

****

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

Arguments:

lb As Object
ByVal intRow As Integer
ByVal intCol As Integer
intItems As Integer
Optional ByVal intDirection As Integer
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional objSelectItems As Object
Optional blnCellValue As Boolean

Returns: Short

 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 Short
'select multiple items
Dim arr() As String = {"pink floyd", "love", "cream"}
'select one item
'Dim arr As String = "pink floyd"
ListBox1.Items.Clear()
i = oExcel.ExcelToListBox(ListBox1, 1, 1, 20, DirectionType.xloDown, False, True, arr)
```

****

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

Arguments:

lv As Object
ByVal intRow As Integer
ByVal intCol As Integer
intItems As Integer
Optional ByVal intDisplay As Integer
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional objSelectItems As Object
Optional blnCellValue As Boolean

Returns: Short

 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 Short
Dim arr(1) As String
 arr(0) = "Fred Brown"
 arr(1) = "Karen Jones"
 ListView1.Items.Clear()
 i = oExcel.ExcelToListView(ListView1, 1, 6, 10, DisplayType.xloNormal, False, True, arr)
```

****

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

Arguments:

objArray As Object
ByVal intRow As Integer
ByVal intCol As Integer
intItems As Integer
Optional ByVal intDirection As Integer
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional blnCellValue As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.ExcelToOneDArray(arr, 1, 1, 10,  DirectionType.xloDown, True, False)
For i = 0 To UBound(arr)
   MsgBox arr(i)
Next
```

****

ExcelToADORecordset() - Function to copy Excel worksheet cells to an ADO recordset.

Arguments:

objConn As Object
objRecd As Object
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnOverwrite As Boolean

Optional blnCellValue As Boolean

Returns: Short

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

Example:

```
    Dim i As Short
    Dim strConn As String
    Dim cn As New ADODB.Connection()
    Dim rs As New ADODB.Recordset()
    Dim s As String
    i = oExcel.OpenWorkbook(Me, AppPath & "\sample2.xls", "sheet2")
    strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & AppPath & _
"\author.mdb;Persist Security Info=False"
    cn.Open(strConn)
    cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
    rs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
    rs.LockType = ADODB.LockTypeEnum.adLockOptimistic
    s = "select * from authors"
    rs.Open(s, cn)
    startTime = Now
    i = oExcel.ExcelToADORecordset(cn, rs, 19, 1, 20, 3, False, True, False)
    StatusBar1.Text = Now.Subtract(startTime).ToString & " seconds"
    oExcel.CloseExcel(False)
    rs.MoveFirst()
    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 created from Excel")
    rs.Close()
    cn.Close()
    rs = Nothing
    cn = Nothing
```

****

ExcelToADORecordset2() - Function to copy Excel worksheet cells to an empty ADO 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: Short

 1 = Success
-1 = Unknown error

Example:

```
Dim i As Integer
 Dim strConn As String
 Dim cn As ADODB.Connection
 Dim rs As ADODB.Recordset
 Dim s As String
 cn = New ADODB.Connection()
 cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
 rs = New ADODB.Recordset()
 rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic
 rs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
 rs.LockType = ADODB.LockTypeEnum.adLockPessimistic
 'Can use range names or Sheet and range
 's = "SELECT * FROM [MyRange]"
 s = "SELECT * FROM [Sheet2$A18:C20]"
 i = oExcel.ExcelToADORecordset2(AppPath & "\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 created from Excel")
 rs.Close()
 cn.Close()
 rs = Nothing
 cn = Nothing
```

****

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

Arguments:

```
tb As Object
ByVal intRow As Integer
ByVal intCol As Integer
intItems As Integer
Optional ByVal intDirection As Integer
Optional blnStopAtEmptyCell As Boolean
Optional blnIgnoreEmptyCell As Boolean
Optional blnCellValue As Boolean
```

Returns: Short

 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 Short
i = OExcel.ExcelToTextBox(TextBox1, 1, 1, 10, DirectionType.xloDown, True, False)

****

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

Arguments:

strTextFile As String
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional ByVal strDelimiter As String
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean

Returns: Short

 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 Short
i = OExcel.ExcelToTextFile("c:\file.txt", 1, 1, 10, 5, ",", False, True)

****

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

Arguments:

objArray As Object
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
intSheets As Short
Optional blnCellValue As Boolean

Returns: Short

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

Example:

```
Dim arr As Object
Dim i As Short
Dim n As Short
Dim j As Short
Dim s As String
i = OExcel.ExcelToThreeDArray(arr, 22, ColType.xlo_F, 26, ColType.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
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
Optional blnCellValue As Boolean
Optional blnExpandAll As Boolean
```

Returns: Short

```
 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 Short
i = OExcel.ExcelToTreeView(TreeView1, 1, ColType.xlo_A, 16, ColType.xlo_D, False, True, False, True)
```

****

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

Arguments:

```
objArray As Object
intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
Optional blnStopAtEmptyRow As Boolean
Optional blnIgnoreEmptyRow As Boolean
```

Optional blnCellValue As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.ExcelToTwoDArray(arr, 1, 1, 10, 5, False, True)
```

C. Miscellaneous functions

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

Arguments:

```
intRow As Integer
intCol As Integer
Optional blnSwitchToExcel As Boolean
```

Returns: Short

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 = OExcel.ActivateCell( 5, ColType.xlo_E, True)
```

****

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

Arguments:

```
Optional objWorkbook As Object
Optional objWorksheet As Object
Optional blnSwitchToExcel As Boolean
```

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ActivateWorkbook("sample1.xls", "Sheet3", True)

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

****

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

Arguments:

objWorksheet As Object
Optional blnSwitchToExcel As Boolean

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ActivateWorksheet("Sheet2", False)

‘The worksheet number can also be used
i = OExcel.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 OExcel.ActiveWorkbookName
```

****

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

Arguments: None

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

Example:

```
MsgBox OExcel.ActiveWorkbookPath
```

****

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

Arguments: None

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

Example:

```
MsgBox OExcel.ActiveWorksheetName
```

****

AddWorkbook() - Function to add an Excel workbook.

Arguments:

Optional objTemplate As Object

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.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 intInsert As Integer
Optional objWorkbook As Object

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.AddWorksheet("My New Sheet", InsertType.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: Short

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

Examples:

```
Dim i As Short

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

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

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

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

****

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

Arguments:

strRange As String
strCharacters As String

Returns: Short

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.AutoCompleteCells("A1:A10", "Be")
i = OExcel.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 intFillType As Integer

Returns: Short

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

Example:

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

****

CalculateWorkbook() - Function to calculate an Excel workbook.

Arguments:

Optional objWorkbook As Object

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.CalculateWorkbook("MyWorkbook.xls")
```

****

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

Arguments: None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.CalculateWorkbooks
```

****

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

Arguments:

intCenter As Integer
blnCenter As Boolean

Returns: Short

 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 Short
i = OExcel.CenterPage(CenterType.xloCenterVertically, True)
```

****

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

Arguments:

None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.CheckSpelling
```

****

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

Arguments:

cl As Collection
objArray As Object

Returns: Short

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

Example:

```
Dim i As Short
Dim Temperatures As Collection
Dim arr As Object
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 = OExcel.CollectionToOneDArray(Temperatures, arr)
Set Temperatures = Nothing 'Destroy Collection Object
```

****

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

cb As Object
objArray As Object

Returns: Short

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

Example:

```
Dim i As Short
```

```
Dim arr As Object
i = OExcel.ComboBoxToOneDArray(Combo1, arr)
```

****

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

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ClearCellComments("A1:E5")
```

****

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

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ClearCellFormats("A1:E5")
```

****

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

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ClearCells("A1:E5")
```

****

CloseExcel() - Function to close Excel.

Arguments:

blnSave As Boolean

Returns: None

Example:

OExcel.CloseExcel True

****

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

Arguments:

blnSave As Boolean

Returns: Short

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

Example:

Dim i As Short
i = OExcel.CloseWorkbook(True)

****

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

Arguments:

intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer
intCompute As Integer
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 intCompute (valid values are 0,1,2,3)
-5 = Unknown error

Example:

Dim i As Short
i = OExcel.ComputeCells(1, ColType.xlo_M, 5, ColType.xlo_N, ComputeType.xloAdd, 1.5, True, True)

****

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

Arguments:

objArray As Object
intCompute As Integer
dblNumber As Double

Returns: Short

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

Example:

```
Dim i As Short
Dim arr(10) As Object
For i = LBound(arr) To UBound(arr)
    arr(i) = i
Next
i = OExcel.ComputeOneDArray(arr, ComputeType.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:

objArray As Object
intCompute As Integer
dblNumber As Double

Returns: Short

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

Example:

```
Dim arr(0 To 4, 0 To 4, 0 To 2) As Object
Dim i As Short
Dim n As Short
Dim j As Short
Dim k As Short
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 = OExcel.ComputeThreeDArray(arr, ComputeType.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:

objArray As Object
intCompute As Integer
dblNumber As Double

Returns: Short

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

Example:

```
Dim arr(0 To 9, 0 To 9) As Object
Dim i As Short
Dim n As Short
Dim k As Short
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 = OExcel.ComputeTwoDArray(arr, ComputeType.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 intPasteType As Integer
Optional intPasteSpecialOperation As Integer
Optional blnSkipBlanks As Boolean
Optional blnTranspose As Boolean
Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.CopyRange("A1:A10", "D1:D10", PasteType.xloPasteAll,
PasteSpecialOperation.xloPasteSpecialOperationNone, False, False)
```

****

CreateChart() - Function to create a chart in Excel

Arguments:

strRange As String
Optional intChartType As Integer
Optional ByVal intSeriesType As Integer
Optional ByVal intLocationType As Integer
Optional ByVal intTop As Short
Optional ByVal intLeft As Short
Optional strSheet As String
Optional strTitle As String
Optional strCategoryTitle As String
Optional strValueTitle As String
Optional strExtraTitle As String

Returns: Short

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

Example:

```
Dim i As Short
'Create a chart within sheet
i = OExcel.CreateChart("B5:C50", ChartType.xlo3DColumn, SeriesType.xloRows,
LocationType.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: Short

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

Example:

```
Dim i As Short
i = OExcel.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 Short = 10
Optional intRotation As Short
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: Short

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

Example:

```
Dim i As Short
'Open a workbook and activate Sheet1
i = OExcel.OpenWorkbook(Me, Me, AppPath & "\sample2.xls", "Sheet1", False)
'Create a PDF file from the active worksheet
'Note: valid font names are Arial, Courier, and Times-Roman
i = OExcel.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
OExcel.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 Short = 10
Optional intRotation As Short
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: Short

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

Example:

```
Dim i As Short
Dim strFile As String
strFile = AppPath & "\pdf.txt"
'Open a workbook and activate Sheet1
i = OExcel.OpenWorkbook(Me, AppPath & "\sample2.xls", "Sheet1", False)
'Create a text file from the active worksheet
i = OExcel.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 = OExcel.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
OExcel.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: Short

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

Example:

```
Dim i As Short
'Open a workbook and activate Sheet3
i = OExcel.OpenWorkbook(Me, AppPath & "\sample2.xls", "Sheet3", False)
'Create a PDF file from the specified active worksheet range
i = OExcel.CreatePDFUsingDistiller(AppPath & "\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, AppPath & "\test.pdf", vbNullString, vbNullString, 1
  End If
Else
  MsgBox "Unable to create PDF file.", vbInformation
End If
OExcel.CloseExcel False
```

****

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

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

Returns: Short

```
 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 Short
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 = OExcel.OpenWorkbook(Me, AppPath & "\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 = OExcel.CreatePivotTable("A9:C25", "Sheet3!R1C5", RowFields, ColFields, PivotFields, "MyPivotTable")
'line below creates a pivot table in a new sheet
'i = OExcel.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 Object
strTargetRange As String
arrRowFields As Object
arrColumnFields As Object
arrPivotFields As Object
Optional strTableName As String
Optional blnRowTotals As Boolean
Optional blnColumnTotals As Boolean
Optional blnSaveData As Boolean

Returns: Short

 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 Short
Dim strConn As String
Dim strSQL As String
Dim arr As Object
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=" & AppPath & "\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 = OExcel.OpenWorkbook(Me, AppPath & "\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 = OExcel.CreatePivotTableFromDB(arr, "Sheet3!R1C5", RowFields, ColFields, PivotFields, "MyDBPivotTable")
'line below creates a pivot table in a new sheet
'i = OExcel.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: Short

 1 = Success
-1 = Unknown error

Example:

```
Dim strFile As String
Dim i As Short
strFile = AppPath & "\MyFile.xls"
i = OExcel.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: Short

 1 = Success
-1 = Unknown error

Example:

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

****

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

Arguments:

strDSN As String
strSQL As String
objArray As Object
Optional strUserID As String
Optional strPassword As String
Optional blnIncludeHeadings As Boolean

Returns: Short

 1 = Success
-1 = objArray 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 Short
Dim arr As Object
i = OExcel.ADODatabaseTableToArray("MyDSN","SELECT * FROM MyTable",arr,"","",False)
```

****

ADODatabaseTableToTextFile() - 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: Short

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

Example:

```
Dim i As Short
'Create a comma-delimited text file from a recordset
i = OExcel.ADODatabaseTableToTextFile("MyDSN", "Select * From Authors", c:\Authors.csv", "", "", True,",")
'Open the comma-delimited text file in Excel
i = OExcel.OpenWorkbook(Me, "c:\Authors.csv")
```

****

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

Arguments:

```
dg As Object
objArray As Object
Optional blnIncludeHeadings As Boolean
```

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.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.

****

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

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.DeleteRange("A1:D10")
```

****

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

Arguments:

Optional objWorksheet As Object
Optional objWorkbook As Object
Optional blnConfirmMessage As Boolean
Optional strFormCaption As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.DeleteWorksheet("Sheet2", , False, Me.Caption)

'The worksheet number can also be used
i = OExcel.DeleteWorksheet(2, , False, Me.Caption)
```

****

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

Arguments:

dt As Object
objArray As Object
Optional blnIncludeKeys As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim vItem As Object
Dim vKey As Object
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 = OExcel.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
objArray As Object
```

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.DirListBoxToOneDArray(Dir1, arr)
```

****

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

Arguments: None

```
dlb As Object
objArray As Object
```

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.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
objRecipients As Object
Optional strSubject As String
Optional blnReturnReceipt As Boolean

Returns: Short

 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 Short
Dim Recipients(1) As String
Recipients(0) = "john.doe@mail.com"
Recipients(1) = "jane.doe@mail.com"
i = OExcel.EmailWorkbook("MailServerUsername", "MailServerPassword", Recipients, "My Workbook", False)
'If you need to email to just one recipient simply pass a string:
'i = OExcel.EmailWorkbook("MailServerUsername", "MailServerPassword", "john.doe@mail.com", "My Workbook", False)

****

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

Arguments: None

Returns: Integer Integer

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

Example:

MsgBox OExcel.ExcelsFreeMemory

****

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

Arguments: None

Returns: Integer Integer

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

Example:

MsgBox OExcel.ExcelsTotalMemory

****

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

Arguments: None

Returns: Integer Integer

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

Example:

MsgBox OExcel.ExcelsUsedMemory

****

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

Arguments:

strRange As String
strTextToFind As String
Optional blnCaseSensitive As Boolean
Optional intCellData As Integer
Optional intCellPart As Integer
Optional strCell As String

Returns: Short

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 Short
Dim strCellAddress As String
strSearchFor = "FLOYD"
strStartRange = "A1:D6"
i = OExcel.FindInRange(strStartRange, strSearchFor, False, CellDataType.xloValues,
CellPartType.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 intPattern As Integer
Optional intCellData As Integer
Optional intCellPart As Integer
Optional objCells As Object

Returns: Integer 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 Integer
strSearchFor = "EA"
i = OExcel.FindInRangeAndHighlight("A1:D6", strSearchFor, False, PatternType.xloPatternGray16,
CellDataType.xloValues, CellPartType.xloPart)
If i > 0 Then
  MsgBox "Text '" & strSearchFor & "' found. " & i & " cells highlighted."
  'Now remove pattern from cells
  i = OExcel.FindInRangeAndHighlight("A1:D6", strSearchFor, False, PatternType.xloPatternNone,
CellDataType.xloValues, CellPartType.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 intCellData As Integer
Optional intCellPart As Integer
Optional blnReplaceAll As Boolean
Optional intPattern As Integer
Optional objCells As Object
```

Returns: Integer 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 Integer
strSearchFor = "EA"
strReplaceWith = "XX"
i = OExcel.FindInRangeAndReplace("A1:D6", strSearchFor, strReplaceWith, False, CellDataType.xloValues,
CellPartType.xloPart, False, PatternType.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 intCellData As Integer
Optional intCellPart As Integer
Optional strCell As String

Returns: Short

 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 Short
strSearchFor = "FLOYD"
i = OExcel.FindInWorksheet(strSearchFor, False, CellDataType.xloValues, CellPartType.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 intPattern As Integer
Optional intCellData As Integer
Optional intCellPart As Integer
Optional objCells As Object

Returns: Integer 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 Integer
strSearchFor = "EA"
i = OExcel.FindInWorksheetAndHighlight(strSearchFor, False, PatternType.xloPatternGray16,
CellDataType.xloValues, CellPartType.xloPart)
If i > 0 Then
   MsgBox "Text '" & strSearchFor & "' found. " & i & " cells highlighted."
   'Now remove pattern from cells
   i = OExcel.FindInWorksheetAndHighlight(strSearchFor, False, PatternType.xloPatternNone,
CellDataType.xloValues, CellPartType.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 intCellData As Integer
Optional intCellPart As Integer
Optional blnReplaceAll As Boolean
Optional intPattern As Integer
Optional objCells As Object
```

Returns: Integer 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 Integer
strSearchFor = "EA"
strReplaceWith = "XX"
i = OExcel.FindInWorksheetAndReplace(strSearchFor, strReplaceWith, False, CellDataType.xloValues,
CellPartType.xloPart, False, PatternType.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
objArray As Object
Optional blnSelectedOnly As Boolean
```

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.FileListBoxToOneDArray(File1, arr, True)
```

****

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

Arguments:

fg As Object
objArray As Object
Optional intStartRow As Integer
Optional intStartCol As Integer
Optional ByVal intEndRow As Integer
Optional ByVal intEndCol As Integer

Returns: Short

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

Example:

```
Dim arr As Object
Dim i As Short
Dim n As Short
Dim c As String
Screen.MousePointer = vbHourglass
i = OExcel.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:

intRow As Integer
intCol As Integer
Optional intScrollRow As Integer
Optional intScrollCol As Integer

Returns: Short

 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 Short
‘freeze pane at row 5, column 5
i = OExcel.FreezePanes(5, ColType.xlo_E)
```

****

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

Arguments:

objExcelApp As Object

Returns: Short

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

Example:

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

****

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

Arguments:

objRange As Object
strRange As String

Returns: Short

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

Example:

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

****

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

Arguments:

objArray As Object

Returns: Short

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

Example:

```
Dim arr As Object
Dim i As Short
```

```
Dim n As Short
Dim strBook As String
n = OExcel.OpenWorkbook(Me, AppPath & "\sample1.xls")  'open workbook
n = OExcel.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 = OExcel.ActivateWorkbook(strBook)  'activate the workbook
   MsgBox strBook, vbOKOnly, "Workbook Name"
Next
OExcel.CloseExcel (False)  'close Excel without saving changes
```

****

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

Arguments:

objExcelWorkbook As Object

Returns: Short

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

Example:

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

****

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

Arguments:

objArray As Object

Returns: Short

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

Example:

```
Dim arr As Object
Dim i As Short
Dim n As Short
Dim strSheet As String
n = OExcel.OpenWorkbook(Me, AppPath & "\sample2.xls")  'open workbook
n = OExcel.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 = OExcel.ActivateWorksheet(strSheet)  'activate the worksheet
   n = OExcel.SetPattern("A1:A5", PatternType.xloPatternGray16)  'set a pattern for a range of cells
   MsgBox strSheet, vbOKOnly, "Worksheet Name"
```

```
Next
OExcel.CloseExcel (False)  'close Excel without saving changes
```

****

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

Arguments:

objExcelWorksheet As Object

Returns: Short

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

Example:

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

****

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

Arguments:

intCol As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.HideColumn(4)
```

****

HideExcel() - Function to hide Excel.

Arguments: None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.HideExcel
```

****

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

Arguments:

intRow As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.HideRow(4)
```

****

HideWorksheet() - Function to hide a worksheet.

Arguments:

Optional objWorksheet As Object
Optional blnVeryHidden As Boolean

Returns: Short

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

Example:

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

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

****

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

Arguments:

intCol As Integer

Returns: Short

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

Example:

```
Dim i As Short
Dim n As Integer
n = 4
i = OExcel.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: Short

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

Example:

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

****

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

Arguments:

```
objArray As Object
ByVal intRow As Integer
intCol As Integer
```

Returns: Short

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

Example:

```
Dim i As Short
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 = OExcel.InsertHyperlinks(arr, 1, ColType.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: Short

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

Example:

Dim i As Short
‘Note: Passed numbers for Left, Top, Width, and Height indicate pixels
i = OExcel. InsertLinkedPicture(AppPath & "\file.jpg", 100, 50, 200, 150)

****

InsertPageBreak() – Function to insert a page break into the active Word document.

Arguments: None

Returns: Short

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

Example:

Dim i As Short
i = OExcel.InsertPageBreak

****

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: Short

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

Example:

```
Dim i As Short
‘Note: Passed numbers for Left, Top, Width, and Height indicate pixels
i = OExcel. InsertPicture(AppPath & "\file.jpg", 100, 50, 200, 150)
```

****

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

Arguments:

intRow As Integer

Returns: Short

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

Example:

```
Dim i As Short
Dim n As Integer
n = 4
i = OExcel.InsertRow(n)
```

****

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

Arguments:

intRow As Integer
intCol As Integer

Returns: Short

```
 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 Short
i = OExcel.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:

intStartRow As Integer
intStartCol As Integer
intEndRow As Integer
intEndCol As Integer

Returns: Short

 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 Short
i = OExcel.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 = OExcel.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 OExcel.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
objArray As Object
Optional blnSelectedOnly As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.ListBoxToOneDArray(List1, arr, True)
```

****

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

Arguments:

lv As Object
objArray As Object
Optional blnSelectedOnly As Boolean
Optional blnIncludeHeadings As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.ListViewToArray(ListView1, arr, False, True)
```

****

LockRange() - Function to lock a worksheet range.

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.LockRange("A1:A5")
```

****

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

Arguments:

strSourceRange As String
strTargetRange As String
Optional intPasteType As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.MoveRange("A1:A10", "D1:D10", xloPasteAll)
```

****

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

Arguments:

ByVal strText As String
objArray As Object
Optional ByVal strDelimiter As String
Returns: Short

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

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.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: Short

 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 Short
i = OExcel.NameWorksheet("Sheet1", "My Sheet", True)
```

****

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

Arguments:

```
objArray As Object
strText As String
strEndOfRowChar As String
Optional strEndOfArrayChar As String
Optional blnQuotes As Boolean
```

Returns: Short

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

Example:

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

****

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

Arguments:

```
objArray As Object
strTextFile As String
Optional blnQuotes As Boolean
```

Returns: Short

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

Example:

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

****

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

Arguments:

objArray As Object
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean
Optional intTableType As Integer
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Short = 12
Optional intSpaceAfter As Short = 10
Optional intParagraphAlign As Integer
Optional intColumnWidth As Short
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: Short

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

Example:

```
Dim i As Short
Dim arr(0 To 4) As Object
arr(0) = "The Wizard of Oz"
arr(1) = "Dirty Harry"
arr(2) = "Terminator"
arr(3) = "Star Wars"
arr(4) = "2001: A Space Oddysey"
i = OExcel.TextToWord("This table was created from a one dimensional array.", AppPath & "\test.doc", False, False, True, "Arial", 12, 15, xloAlignParagraphLeft)
i = OExcel.OneDArrayToWord(arr, AppPath & "\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:

objForm As System.Windows.Forms.Form
Optional strWorkbook As String
Optional objWorksheet As Object
Optional blnSwitchToExcel As Boolean
Optional strPassword As String
Optional strWriteResPassword As String
Optional strFormCaption As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.OpenWorkbook(Me, AppPath & "\sample1.xls", "Sheet2", True, "openpassword", "writepassword")

‘The worksheet number can also be used
i = OExcel.OpenWorkbook(Me, AppPath & "\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: Short

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

Examples:

```
Dim i As Short

i = OExcel.PrintPreviewRange("A1:D10", True)   'Print preview range & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintPreviewRange("A1:D10", False, Me.Caption)
```

****

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

Arguments:

```
Optional objWorkbook As Object
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String
```

Returns: Short

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

Examples:

```
Dim i As Short
```

```
i = OExcel.PrintPreviewWorkbook( , True)   'Print preview workbook & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintPreviewWorkbook( , False, Me.Caption)
```

****

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

Arguments:

Optional objWorksheet As Object
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Short

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

Examples:

```
Dim i As Short

i = OExcel.PrintPreviewWorksheet("Sheet2", True)   'Print preview worksheet & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintPreviewWorksheet("Sheet2", False, Me.Caption)

‘The worksheet number can also be used
i = OExcel.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: Short

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

Examples:

```
Dim i As Short

i = OExcel.PrintRange("A1:D10", True)   'Print range & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintRange("A1:D10", False, Me.Caption)

****

PrintWorkbook() - Function to print an Excel workbook.

Arguments:

Optional objWorkbook As Object
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Short

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

Examples:

Dim i As Short

i = OExcel.PrintWorkbook(, True)   'Print workbook & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintWorkbook(, False, Me.Caption)

****

PrintWorksheet() - Function to print an Excel worksheet.

Arguments:

Optional objWorksheet As Object
Optional blnSwitchToExcel As Boolean
Optional strFormCaption As String

Returns: Short

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

Examples:

Dim i As Short

i = OExcel.PrintWorksheet("Sheet2", True)   'Print worksheet & switch to Excel
i = OExcel.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 Class .NET method.
i = OExcel.PrintWorksheet("Sheet2", False, Me.Caption)

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

****

ProtectWorkbook() - Function to protect an Excel workbook.

Optional objWorkbook As Object
Optional strPassword As String

Returns: Short

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

Example:

Dim i As Short
i = OExcel.ProtectWorkbook(,"MyPassword")

****

ProtectWorksheet() - Function to protect an Excel worksheet.

Optional objWorksheet As Object
Optional strPassword As String

Returns: Short

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

Example:

Dim i As Short
i = OExcel.ProtectWorksheet (“MySheet”,"MyPassword")

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

****

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

Arguments:

objRecd As Object
objArray As Object
Optional blnIncludeHeadings As Boolean

Returns: Short

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

Example:

```
Dim i As Short
Dim strConn As String
 Dim cn As New ADODB.Connection()
 Dim rs As ADODB.Recordset
```

```
        Dim arr As Object
        Dim n As Integer
        Dim c As String
        strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & AppPath & _
"\author.mdb;Persist Security Info=False"
        cn.Open(strConn)
        cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rs = cn.Execute("Select AU_ID As [ID Number], [Year Born] As [Birth Date], Author As [Author
Name] From Authors")
        i = oExcel.ADORecordsetToArray(rs, arr, True)
        i = oExcel.AdjustColumnWidths("A:C") 'Adjust width of Columns to accomodate widest item
        rs.Close()
        cn.Close()
        rs = Nothing
        cn = Nothing
        GC.Collect()
        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")
```

****

ADORecordsetToTextFile() - 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: Short

 1 = Success
-1 = Unknown error

Example:

```
        Dim strConn As String
        Dim cn As New ADODB.Connection()
        Dim rs As ADODB.Recordset
        Dim arr As Object
        Dim n As Integer
        Dim c As String
        strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & AppPath & _
"\author.mdb;Persist Security Info=False"
        cn.Open(strConn)
        cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
        rs = cn.Execute("Select AU_ID As [ID Number], [Year Born] As [Birth Date], Author As [Author
Name] From Authors")
        'Create tab-delimited text file from a recordset
        i = oExcel.ADORecordsetToTextFile(rs, AppPath & "\authors.txt", True, vbTab, False)
        rs.Close()
        cn.Close()
        rs = Nothing
        cn = Nothing
        GC.Collect()
```

```
    If i = 1 Then
       Shell("notepad.exe " & AppPath & "\Authors.txt", vbNormalFocus)
    End If
```

****

ADORecordsetToWord() - 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 intTableType As Integer
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Short = 12
Optional intSpaceAfter As Short = 10
Optional intParagraphAlign As Integer
Optional intColumnWidth As Short
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: Short

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

Example:

```
    Dim i As Short
    Dim strConn As String
    Dim cn As New ADODB.Connection()
    Dim rs As ADODB.Recordset
    strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & AppPath & _
"\author.mdb;Persist Security Info=False"
    cn.Open(strConn)
    cn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
    rs = cn.Execute("Select AU_ID As [ID Number], [Year Born] As [Birth Date], Author As [Author
Name] From Authors")
    'create a Word table from an ADO recordset
    i = oExcel.TextToWord("This table was created from a recordset. Each column is a fixed width (130
pixels)", AppPath & "\test.doc", False, False, True, "Arial", 15, 15,
ParagraphAlignType.xloAlignParagraphCenter, OrientationType.xloOrientationLandscape)
    i = oExcel.ADORecordsetToWord(rs, True, AppPath & "\test.doc", False, False,
TableType.xloTableFormatColorful2, False, "Verdana", 12, 10,
ParagraphAlignType.xloAlignParagraphRight, 130, True, True, False, True, True, True, False, False,
False, OrientationType.xloOrientationLandscape)
    If i <> 1 Then
       MsgBox("Unable to create document.", vbOKOnly)
    End If
    i = oExcel.InsertPageBreakInWord
    'same table with column widths automatically calculated
```

```
    'note that parameter intColumnWidth = 0 and blnAutoFit = True
    i = oExcel.TextToWord("Same table with column widths automatically calculated.", AppPath &
"\test.doc", False, False, True, "Arial", 15, 15, ParagraphAlignType.xloAlignParagraphCenter,
OrientationType.xloOrientationLandscape)
    i = oExcel.ADORecordsetToWord(rs, True, AppPath & "\test.doc", False, False,
TableType.xloTableFormatColorful2, False, "Verdana", 12, 10,
ParagraphAlignType.xloAlignParagraphCenter, 0, True, True, False, True, True, True, False, False, True,
OrientationType.xloOrientationLandscape)
    If i <> 1 Then
       MsgBox("Unable to create document.", vbOKOnly)
    End If
    i = oExcel.InsertPageBreakInWord
    'same table with column widths automatically calculated to fill entire page width
    'note that parameter intColumnWidth = 0 and blnAutoFit = False
    i = oExcel.TextToWord("Same table with column widths automatically calculated to fill entire page
width.", AppPath & "\test.doc", False, False, True, "Arial", 15, 15,
ParagraphAlignType.xloAlignParagraphCenter, OrientationType.xloOrientationLandscape)
    i = oExcel.ADORecordsetToWord(rs, True, AppPath & "\test.doc", False, False,
TableType.xloTableFormatColorful2, False, "Verdana", 12, 10,
ParagraphAlignType.xloAlignParagraphLeft, 0, True, True, False, True, True, True, False, False, False,
OrientationType.xloOrientationLandscape)
    If i <> 1 Then
       MsgBox("Unable to create document.", vbOKOnly)
    End If
    i = oExcel.InsertPageBreakInWord
    rs.Close()
    cn.Close()
    rs = Nothing
    cn = Nothing
    'create a Word table from a two dimensional array
    Dim arr(3, 2) As Object
    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 = oExcel.TextToWord("This table was created from a two dimensional array.", AppPath &
"\test.doc", False, False, True, "Arial", 12, 15, ParagraphAlignType.xloAlignParagraphLeft,
OrientationType.xloOrientationLandscape)
    i = oExcel.TwoDArrayToWord(arr, AppPath & "\test.doc", False, False,
TableType.xloTableFormatProfessional, False, "Courier", 8, 10,
ParagraphAlignType.xloAlignParagraphCenter, , True, True, False, True, True, True, False, False, True,
OrientationType.xloOrientationLandscape)
    If i <> 1 Then
       MsgBox("Unable to create document.", vbOKOnly)
    End If
    Dim arr2 As Object
    'create a Word table from data in an Excel workbook
    i = oExcel.OpenWorkbook(Me, AppPath & "\sample2.xls", "Sheet1", False)
    i = oExcel.ExcelToTwoDArray(arr2, 12, ColType.xlo_C, 18, ColType.xlo_F, False, True, True)
    oExcel.CloseExcel(False)
    i = oExcel.TextToWord("This table was created from an Excel workbook.", AppPath & "\test.doc",
False, False, True, "Arial", 12, 15, ParagraphAlignType.xloAlignParagraphLeft,
OrientationType.xloOrientationLandscape)
    i = oExcel.TwoDArrayToWord(arr2, AppPath & "\test.doc", False, False,
TableType.xloTableFormat3DEffects2, False, "MS Sans Serif", 12, 10,
```

```
ParagraphAlignType.xloAlignParagraphLeft, , True, True, False, True, True, False, True, False, True,
OrientationType.xloOrientationLandscape)
    'create a Word table from a one dimensional array
    Dim arr3(4) As Object
    arr3(0) = "The Wizard of Oz"
    arr3(1) = "Dirty Harry"
    arr3(2) = "Terminator"
    arr3(3) = "Star Wars"
    arr3(4) = "The Cable Guy"
    i = oExcel.TextToWord("This table was created from a one dimensional array.", AppPath &
"\test.doc", False, False, True, "Arial", 12, 15, ParagraphAlignType.xloAlignParagraphLeft,
OrientationType.xloOrientationLandscape)
    i = oExcel.OneDArrayToWord(arr3, AppPath & "\test.doc", False, False,
TableType.xloTableFormatColorful2, False, "MS Sans Serif", 12, 10,
ParagraphAlignType.xloAlignParagraphCenter, , True, True, False, True, False, False, False, False, True,
OrientationType.xloOrientationLandscape)
    'write a string variable to Word using function TextToWord()
    Dim s As String
    i = oExcel.TwoDArrayToText(arr, s, vbTab, vbCrLf)
    i = oExcel.TextToWord("Text from a two dimensional array.", AppPath & "\test.doc", False, False,
True, "Arial", 12, 15, ParagraphAlignType.xloAlignParagraphLeft,
OrientationType.xloOrientationLandscape)
    i = oExcel.TextToWord(s, AppPath & "\test.doc", False, False, False, "Arial", 10, 0,
ParagraphAlignType.xloAlignParagraphLeft, OrientationType.xloOrientationLandscape)
    'write a string variable to Word using function TextToWord()
    s = ""
    i = oExcel.OneDArrayToText(arr3, s, ", ", ".")
    i = oExcel.TextToWord("Text from a one dimensional array.", AppPath & "\test.doc", False, False,
True, "Arial", 12, 15, ParagraphAlignType.xloAlignParagraphLeft,
OrientationType.xloOrientationLandscape)
    s = "Some of my favorite movies are: " & s
    i = oExcel.TextToWord(s, AppPath & "\test.doc", False, False, False, "Arial", 10, 0,
ParagraphAlignType.xloAlignParagraphLeft, OrientationType.xloOrientationLandscape)
    MsgBox("Document created.", vbOKOnly, "Document")
```

****

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

Arguments:

None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ResetPageBreaks
```

****

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

Arguments:

intMacro As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.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: Short

 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 Short
i = OExcel.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 Short
i = OExcel.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: Short

1 = Success
-1 = Excel is not active

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

Example:

Dim i As Short
i = OExcel. SetBackgroundPicture (AppPath & “\file.jpg”)

****

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

Arguments:

blnBlackAndWhite As Boolean

Returns: Short

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

Example:

Dim i As Short
i = OExcel.SetBlackAndWhite(True)

****

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

Arguments:

strRange As String
ByVal intBorderIndex As Integer
Optional intColor As Integer
Optional intLineStyle As Integer
Optional intWeight As Integer

Returns: Short

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

Example:

Dim i As Short
i = OExcel.SetBorder("A1:A5", xloEdgeRight, RGB(255, 0, 0), xloContinuous, xloMedium)

****

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

Arguments:

strRange As String
ByVal intBorderIndex As Integer
Optional intColor As Integer
Optional intLineStyle As Integer
Optional intWeight As Integer

Returns: Short

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

Example:

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

****

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

Arguments:

```
ByVal intCalculation As Integer
Optional strWorkbook As String
```

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetCalculation(CalcType.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: Short

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

Example:

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

****

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

Arguments:

strRange As String
intColor As Integer

Returns: Short

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

Example:

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

****

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

Arguments:

intCol As Integer
intPageBreak As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetColumnPageBreak(10, xloPageBreakManual)
```

****

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

Arguments:

intCol As Integer
intWidth As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetColumnWidth(5, 50)
```

****

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

Arguments:

strRange As String
strFontName As String

Returns: Short

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

Example:

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

****

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

Arguments:

strRange As String
intColor As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.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 Short

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetFontSize("A1:D10", 10)
```

****

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

Arguments:

strRange As String
ByVal intFontSpecial As Integer

Returns: Short

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

Example:

Dim i As Short
i = OExcel.SetFontSpecial("A1:D10", FontSpecialType.xloSuperscript)

****

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

Arguments:

strRange As String
ByVal intFontStyle As Integer

Returns: Short

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

Example:

Dim i As Short
i = OExcel.SetFontStyle("A1:D10", FontType.xloBold)

****

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

Arguments:

strRange As String
intFontUnderline As Integer

Returns: Short

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

Example:

Dim i As Short
i = OExcel.SetFontUnderline("A1:D10", UnderlineType.xloUnderlineStyleSingle)

****

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

Arguments:

strFooter As String
intFooter As Integer

Returns: Short

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

Example:

```
Dim i As Short
'Display date, time, page number, and custom string in footer
i = OExcel.SetFooter("&D &T &P This is a right-aligned footer.", FooterType.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:

bintridLines As Boolean

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetGridLines(True)
```

****

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

Arguments:

strHeader As String
intHeader As Integer

Returns: Short

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

Example:

```
Dim i As Short
'Display date, time, page number, and custom string in header
i = OExcel.SetHeader("&D &T &P This is a right-aligned header.", HeaderType.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: Short

 1 = Success
-1 = Excel is not active

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

Example:

```
Dim i As Short
i = OExcel.SetHeadings(True)
```

****

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

Arguments:

```
intMargin As Integer
dblInches As Double
```

Returns: Short

```
 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 Short
i = OExcel.SetMargin(MarginType.xloMarginRight, 0.5)
```

****

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

Arguments:

```
intOrientation As Integer
```

Returns: Short

```
 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 Short
i = OExcel.SetOrientation(OrientationType.xloOrientationLandscape)  'Use landscape mode
```

****

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

Arguments:

```
strRange As String
intPattern As Integer
```

Returns: Short

```
 1 = Success
```

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

Example:

```
Dim i As Short
i = OExcel.SetPattern("A1:D10", PatternType.xloPatternGray50)
```

****

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

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetPrintArea("A1:D10")
```

****

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

Arguments:

intRow As Integer
intHeight As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetRowHeight(10, 20)
```

****

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

Arguments:

intRow As Integer
intPageBreak As Integer

Returns: Short

 1 = Success

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

Example:

```
Dim i As Short
i = OExcel.SetRowPageBreak(10, PageBreakType.xloPageBreakManual)
```

****

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

Arguments:

strRange As String
intTextHorizontal As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetTextHorizontal("A1:D10", HorizontalTextType.xloHAlignCenter)
```

****

SetTextVertical() - Function to set the vertical text alignment for a worksheet range.

Arguments:

strRange As String
intTextVertical As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.SetTextVertical("A1:D10", VerticalTextType.xloVAlignCenter)
```

****

SetTitle() - Function to set the printed worksheet page's row/column titles.

Arguments:

intTitle As Integer
strTitleRange As String

Returns: Short

 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 Short
i = OExcel.SetTitle(TitleType.xloTitleColumn, "A1:A5")
```

****

SetZoom() - Function to zoom a printed worksheet - 10% to 400%.

Arguments:

intZoomPercent As Short

Returns: Short

 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 Short
i = OExcel.SetZoom(150)
```

****

SortRange() - Function to sort a range of Excel cells. Can sort on up to three columns.

Arguments:

strRange As String
objField1 As Object
Optional intSortOrder1 As Integer
Optional objField2 As Object
Optional intSortOrder2 As Integer
Optional objField3 As Object
Optional intSortOrder3 As Integer

Returns: Short

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

Example:

```
Dim i As Short
Dim objRange1 As Object
Dim objRange2 As Object
i = OExcel.GetRangeObject(objRange1, "A1")
i = OExcel.GetRangeObject(objRange2, "B1")
i = OExcel.SortRange("A1:C20", objRange1, SortOrderType.xloAscending, objRange2,
SortOrderType.xloAscending)
```

****

SplitWindow() – Function to split the active worksheet window at the specified row/column.

Arguments:

intRow As Integer
intCol As Integer

Returns: Short

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

Example:

```
Dim i As Short
‘split worksheet window at row 5, column 5
i = OExcel.SplitWindow(5, ColType.xlo_E)
```

****

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

Arguments:

tb As Object
objArray As Object

Returns: Short

1 = Success
-1 = Object is not a TextBox
-2 = objArray is not an array
-3 = Unknown error

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.TextBoxToOneDArray(Text1, arr)
```

****

TextFileToArray() - Function to copy text file contents to an array.

Arguments:

strTextFile As String
objArray As Object
Optional ByVal strDelimiter As String
Optional blnRemoveQuotes As Boolean

Returns: Short

1 = Success
-1 = Text file not found
-2 = objArray is not an array
-3 = Unknown error

Example:

```
Dim i As Short
Dim arr As Object
i = OExcel.TextFileToArray("c:\file.txt", arr, ",")
```

****

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

Arguments:

```
strTextFile As String
objArray As Object
```

Returns: Short

```
 1 = Success
-1 = Text file not found
-2 = objArray is not an array
-3 = Unknown error
```

Example:

```
Dim arr As Object
Dim i As Short
i = OExcel.TextFileToOneDArray(AppPath & "\file.txt", arr)
i = OExcel.OneDArrayToExcel(arr, 1, 1, DisplayType.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 Short = 12
Optional intSpaceAfter As Short = 10
Optional intParagraphAlign As Integer
```

Returns: Short

```
 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
objArray As Object
Optional blnSuppressRows As Boolean
```

Returns: Short

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

Example:

```
'This example assumes a TreeView control is on your form
Dim i As Short
Dim n As Integer
Dim arr As Object
Dim s As String
i = oExcel.TreeViewToTwoDArray(TreeView1, arr, False)
For i = 0 To UBound(arr, 1)
   For n = 0 To UBound(arr, 2)
      s = s & arr(i, n) & vbTab
   Next
   s = s & vbCrLf
Next
MsgBox(s, vbOKOnly, "Array Items")
```

****

TwoDArrayToCollection() - Function to copy a two dimensional array to a collection object.

Arguments

objArray As Object
cl As Collection

Returns: Short

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

Example:

```
Dim i As Short
Dim objCol As Collection
Dim s As String
Dim arr(2, 1) As Object
'Create collection object
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 = oExcel.TwoDArrayToCollection(arr, objCol)
If i <> 1 Then Exit Sub
For i = 1 To objCol.Count
   s = s & objCol.Item(i) & vbCrLf
Next
MsgBox(s, vbOKOnly, "Collection Items")
'Get a specific collection item
MsgBox("Index 2 = " & objCol.Item(2), vbOKOnly, "Specific Item")
```

```
objCol = Nothing     'Destroy Collection Object
```

****

TwoDArrayToDictionary() - Function to copy a two dimensional array to a dictionary object.

objArray As Object
dt As Object

Returns: Short

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

Example:

```
Dim arr() As String
Dim i As Short
Dim arr2(2, 1) As Object
Dim objDict As New MyDictionary()
Dim s As String
Dim vItem As Object
Dim vKey As Object
i = 0
'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 = oExcel.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.Values
   arr(i) = arr(i) & ", Item: " & vItem
   i = i + 1
Next
For i = 0 To objDict.Count - 1
   s = s & arr(i) & vbCrLf
Next
MsgBox(s, vbOKOnly, "Dictionary Items")
'Get a specific dictionary item
MsgBox("Key 2 = " & objDict.Item(2), vbOKOnly, "Specific Item")
objDict = Nothing     'Destroy Dictionary Object
```

****

TwoDArrayToFlexGrid() - Function to copy a two dimensional array to a FlexGrid control.

objArray As Object
fg As Object

Returns: Short

 1 = Success
-1 = objArray 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 Object
Dim i As Short
Dim n As Short
Dim j As Short
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 = OExcel.TwoDArrayToFlexGrid(arr, MSFlexGrid1)
```

****

TwoDArrayToListView() - Function to copy a two dimensional array to a ListView control.

```
objArray As Object
lv As Object
Optional objSelectItems As Object
```

Returns: Short

 1 = Success
-1 = objArray 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 Short
Dim arr(3, 2) As Object
'Dim arr2() As String = {"Billy Bob", "Wayne Cohen"}
Dim arr2(1) As String
arr2(0) = "billy bob"
arr2(1) = "wayne cohen"
'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"
ListView1.Items.Clear()
i = oExcel.TwoDArrayToListView(arr, ListView1, arr2)
```

****

TwoDArrayToText() - Function to copy a two dimensional array to a string variable.

Arguments:

objArray As Object
strText As String
strEndOfRowChar As String
Optional strEndOfArrayChar As String
Optional blnQuotes As Boolean

Returns: Short

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

Example:

```
Dim arr(0 To 3, 0 To 2) As Object
Dim s As String
Dim i As Short
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 = OExcel.TwoDArrayToText(arr, s, vbTab, vbCrLf, True)
MsgBox s
```

****

TwoDArrayToTextFile() - Function to copy a two dimensional array to a text file.

Arguments:

objArray As Object
strTextFile As String
Optional ByVal strDelimiter As String
Optional blnQuotes As Boolean

Returns: Short

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

Example:

```
Dim arr(0 To 10, 0 To 9) As Object
Dim i As Short
Dim n As Short
Dim j As Short
```

```
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 = OExcel.TwoDArrayToTextFile(arr, AppPath & "\TwoD.csv", ",", True)
Screen.MousePointer = vbDefault
'Open the comma-delimited text file in Notepad
Shell "notepad.exe " & AppPath & "\TwoD.csv", vbNormalFocus
```

****

TwoDArrayToTreeView() – Function to copy a two dimensional array to a TreeView control.

Arguments:

objArray As Object
tv As Object
Optional blnExpandAll As Boolean

Returns: Short

1 = Success
-1 = objArray 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 Short
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 = OExcel.TwoDArrayToTreeView(arr, TreeView1, True)
```

****

TwoDArrayToWord() - Function to copy a two dimensional array to a MS Word document.

Arguments:

objArray As Object
Optional strDocument As String
Optional blnSave As Boolean
Optional blnClose As Boolean

Optional intTableType As Integer
Optional blnBold As Boolean
Optional strFont As String = "Times New Roman"
Optional intSize As Short = 12
Optional intSpaceAfter As Short = 10
Optional intParagraphAlign As Integer
Optional intColumnWidth As Short
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: Short

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

Example:

```
Dim i As Short
Dim arr(0 To 3, 0 To 1) As Object
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 = OExcel.TextToWord("This table was created from a two dimensional array.", AppPath & "\test.doc", False, False, True, "Arial", 12, 15, xloAlignParagraphLeft)
i = OExcel.TwoDArrayToWord(arr, AppPath & "\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: Short

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

Example:

```
Dim i As Short
i = OExcel.UnfreezePanes
```

****

UnhideColumn() - Function to unhide a column in a worksheet.

Arguments:

intCol As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnhideColumn(4)
```

****

UnhideExcel() - Function to unhide Excel.

Arguments: None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnhideExcel
```

****

UnhideRow() - Function to unhide a row in a worksheet.

Arguments:

intRow As Integer

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnhideRow(4)
```

****

UnhideWorksheet() - Function to unhide a worksheet.

Arguments:

Optional objWorksheet As Object

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnhideWorksheet("My Worksheet")

‘The worksheet number can also be used
i = OExcel.UnhideWorksheet(1)
```

****

UnlockRange() - Function to unlock a worksheet range.

Arguments:

strRange As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnlockRange("A1:A5")
```

****

UnprotectWorkbook() - Function to unprotect an Excel workbook.

Arguments:

Optional objWorkbook As Object
Optional strPassword As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnprotectWorkbook( , "MyPassword")
```

****

UnprotectWorksheet() - Function to unprotect an Excel worksheet.

Arguments:

Optional objWorksheet As Object
Optional strPassword As String

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnprotectWorksheet(“MySheet”, "MyPassword")

‘The worksheet number can also be used
i = OExcel.UnprotectWorksheet(1, "MyPassword")
```

****

UnsplitWindow() – Function to unsplit the active worksheet window.

Arguments: None

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.UnsplitWindow
```

****

WorksheetCount() – Function that returns the number of worksheets for the active workbook.

Arguments:

None

Returns: Integer Integer

= Success - number of worksheets in the active workbook
-1 = Excel is not active
-2 = Excel workbook not active

Example:

```
Dim i As Short
Dim n As Short
For i = 1 To OExcel.WorksheetCount
   n = OExcel.ActivateWorksheet(i)  'activate the worksheet
   n = OExcel.SetPattern("A1:A5", PatternType.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
objArray As Object
Optional blnIncludeHeadings As Boolean

Returns: Short

 1 = Success
-1 = objArray is not an array
-2 = Unable to access XML DLL
-3 = Unable to open XML file
-4 = Unknown error

Example:

```
Dim arr As Object
Dim i As Short
Dim n As Short
Dim s As String
i = OExcel.XMLFileToArray(AppPath & "\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"
```

****

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: Short

 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 Short
TreeView1.Nodes.Clear
i = OExcel.XMLFileToTreeView(AppPath & "\file.xml", TreeView1, False)
'To use an XML file located on a web server use the URL
'i = OExcel.XMLFileToTreeView("http://www.mypage.com/file.xml", TreeView1, True)
```

****

ZoomWorksheet() - Function to zoom in/out of a worksheet - 10% to 400%.

Arguments:

intZoomPercent As Short

Returns: Short

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

Example:

```
Dim i As Short
i = OExcel.ZoomWorksheet(200)
```