ACTIVEREPORTS' EXPORT FILTERS

ActiveReports gives you the option to export in different formats by providing modular COM DLLs. The ActiveReports export filters include an Excel export filter, a RTF export filter, a PDF export filter, and a text export filter. You can use any of these filters to export your reports to Excel file format, Rich Text Format (RTF), Portable Document Format (PDF) and to a delimited text file. There are four steps to using the export filters in your project.

1. Reference the filter in your project using the Project/References dialog box.
2. Create the export object.
3. Set the export filename.
4. Pass an ActiveReports pages collection to the export object's Export method.

Using the Export Filters

Example 1

In this example, this code is placed in the ReportEnd Event of an ActiveReports Designer, use it anywhere you can pass the Export method a valid ActiveReports pages collection.

```
Private Sub ActiveReport_ReportEnd()
    Dim myExportObject As ActiveReportsExcelExport.ARExportExcel

    Set myExportObject =
CreateObject("ActiveReportsExcelExport.ARExportExcel")
    myExportObject.FileName = App.Path & "\XLReport.xls"
    myExportObject.Export Me.Pages
    Set myExportObject = Nothing
End Sub
```

Example 2

In this example, we have added a reference to the "ActiveReports Excel Export Filter" library through VB's References Command under the Project menu.

```
Private Sub ActiveReport_ReportEnd()
    Dim myExportObject As ActiveReportsExcelExport.ARExportExcel

    Set myExportObject = New ActiveReportsExcelExport.ARExportExcel
    myExportObject.FileName = App.Path & "\XLReport.xls"
    myExportObject.Export Me.Pages
    Set myExportObject = Nothing
End Sub
```

It's easy to make the mistake of passing an empty pages collection to the export. Make sure that you run your report before you try to export from it.

The ActiveReports pages collection has a native file format called RDF. It's often useful to save pages, also called canvases, to this format, and export them later.

```
Private Sub ActiveReport_ReportEnd()
   ' save the pages collection
   Me.Pages.Save App.Path & "\RDFReport.RDF"
End Sub
```

You can use an ActiveReports Viewer control placed on a form to load and export the report.rdf file.

```
Private Sub cmdARVExport_Click()
    Dim myExportObject As ActiveReportsPDFExport.ARExportPDF
    Set myExportObject = New ActiveReportsPDFExport.ARExportPDF

    ARViewer21.Pages.Load App.Path & "\RDFReport.RDF"
    myExportObject.FileName = App.Path & "\PDFReport.PDF"
    myExportObject.Export ARViewer21.Pages
```

```
    Set myExportObject = Nothing
End Sub
```

Distributing Export Filters

When using any of the export filters in your project, you should include the associated DLL files in your application distribution.

PDFEXPT.DLL	Portable Document Format (PDF) Export Filter
RTFEXPT.DLL	Rich Text Format (RTF) Export Filter
TEXTEXPT.DLL	Text Export Filter
EXCLEXPT.DLL	Microsoft Excel Export Filter
HTMLEXPT.DLL	Hyper Text Markup Language (HTML) Export Filter
TIFFEXPT.DLL	TIFF Export Filter

These files are COM components and must be registered on the target system when installing your application.

EXCEL EXPORT FILTER

What is the ActiveReports Excel Export Filter?

The ActiveReports Excel Export Filter is a DLL based library that exports an ActiveReports pages collection to Microsoft Excel. The export filter can generate single or multiple sheet workbooks, with control over generation of white space to match the printable report. The export has support for ActiveReports borders, lines and pictures, and will attempt to map ActiveReports border and line styles to excel styles intelligently depending on Excel output version. The export automatically generates number formats and has the option of generating horizontal page breaks.

How does the Excel Export Filter work?

ActiveReports Excel Export Filter exports pre-rendered pages from an ActiveReports pages collection. Rendered pages are also called canvases. The export converts from a "forms" model where fields can overlap and stack, to a grid model where only text elements can overlap, and fields share horizontal and vertical boundaries with all other fields on the sheet.

What are the Excel Export Filter's Features?

ActiveReports Excel Export Filter provides two features, the Export Filter, and the Spreadbuilder API.

1. The Export Filter exports a rendered ActiveReports report or pages collection to an excel document.
2. Spreadbuilder's API COM Interfaces allows the creation of--but not loading of--Excel worksheets and workbooks.

The Spreadbuilder API, also in the exclexpt.dll, allows you to build custom spreadsheets cell by cell with fantastic ease of use. The API allows fine control over sheet and workbook layout, with elegantly degrading features across Excel versions 2.1 through Excel 2000. The API allows pictures, styled lines with arrowheads and borders to be added to a virtual sheet, which can be saved multiple times in different versions. The API does not allow the loading of workbooks or sheets.

Registering the Excel Export Library

The ActiveReports Excel Export Filter must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

1. Place the exclexpt.dll file on your machine where you want it in your directory structure.
2. Open the command line by choosing Run from the Start menu.
3. Type the following in the Open box where [filepath] represents the path through the directory structure on your machine to where you have placed the .dll library:
    1. regsvr32 [filepath]/exclexpt.dll
4. A message box indicates that the registration has succeeded.

Adding the Excel Export Filter to your Project

1. Choose **References** from the Project menu.
2. Select *ActiveReports Excel Export Filter 2.0* from the **References** list and click OK.

Using the Excel Export Filter

The code example below follows these steps to export an ActiveReports' Pages Collection to Excel using the ActiveReportsExcelExport object's Export method.

1. Create a new ActiveReportsExcelExport.ARExportExcel object.

```
Dim xls As New ActiveReportsExcelExport.ARExportExcel
Set xls = New ActiveReportsExcelExport.ARExportExcel
```

2. Determine where you want to store the exported information as a .xls document and set the ActiveReportsExcelExport object's FileName property equal to the path and name of that file.

```
sFile = "c:\activereports\vbexport\test1.xls"
xls.FileName = sFile
```

3. Set any other ActiveReportsExcelExport properties you want to specify.
4. Call the ActiveReportsExcelExport object's Export method, passing as a parameter the name of the ActiveReports Pages Collection for export.

```
xls.Export arv.Pages
```

Example:

```
Private Sub mEExpt_Click()
        Dim xls As New ActiveReportsExcelExport.ARExportExcel
        Dim sFile As String
        Dim bSave As Boolean

        Set xls = New ActiveReportsExcelExport.ARExportExcel
        sFile = "c:\activereports\vbexport\test1.xls"
        xls.FileName = sFile
        If arv.Pages.Count > 0 Then
                xls.Export arv.Pages
        End If
End Sub
```

Using the SpreadBuilder API

The SpreadBuilder API allows you to customize the appearance of the Excel files you create. Rather than creating individual sheets or cells, you create a new spreadbuilder object, which represents a workbook containing a collection of sheets. Cells in each sheet are accessed by their row/column address (index is 0-based). You can access individual sheets, rows, columns and cells through the spreadbuilder object. At each level you can access different properties and methods for each object.

Creating a SpreadBuilder Object

1. Set an object variable equal to a new Spreadbuilder object.

```
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
```

2. By default, the workbook is created with one sheet. This translates into a spreadbuilder object containing a Sheets collection consisting of one Sheet.
3. The Sheets collection is zero based. To access the default sheet, use the index number:

```
sb.Sheets(0)
```

4. You can access sheet properties by setting a sheet to an object variable.

```
Dim mysheet as ActiveReportExcelExport.DDSheet
Set mysheet = sb.Sheets(0)
```

5. You can rename the worksheet using the Sheet Name property.

```
      sb.Sheets(0).Name = "Address Book"
mysheet.Name = "Address Book"
```

6. You can add new sheets to the collection using the Sheets Add property.

```
sb.Sheets.Add "AddressBook"
```

7. You can access sheets in the collection by their string name.

```
Set mysheet = sb.Sheets("Sheet1")
```

Example:

```
Private Sub MeExpt_Click()
   Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets.Add "AddressBook2"
End Sub
```

Working with SpreadBuilder sheets

When you have created a spreadbuilder object with sheets, you can begin to customize the design of individual sheets. You can set properties for rows, columns, and cells and designate page breaks.

Accessing the Sheets Collection

Access the Sheets collection through the spreadbuilder object. From the Sheets Collection object you can access individual sheets. The Sheets Collection object exposes six methods.

- Add
- Count
- Item
- Move
- Remove
- Select

Note: *See DDSheets Methods for full documentation.*

Example:

```
sb.Sheets.Add "AddressBook2"
```

Setting Row Properties in SpreadBuilder

To access a row in a spreadbuilder object, open the Sheets collection of the spreadbuilder object to the sheet and the row you need. The Row object exposes two properties:

- Autosize
- Height

Note: *See DDRow Properties for full documentation.*

You cannot use these two properties together in the same Row. Setting the Height property automatically sets the Autosize property to FALSE.

Example:

```
sb.Sheets(0).Rows(0).AutoSize = True
```

Setting Column Properties in SpreadBuilder

To access a column in a spreadbuilder object, open the Sheets collection of the spreadbuilder object to the sheet and the column you need. The Column object exposes one property:

- Width

Note: *See DDColumn Properties for full documentation*

Example:

```
sb.Sheets(0).Columns(0).Width = 800
```

Setting Cell Properties in SpreadBuilder

To access a cell in a spreadbuilder object, open the Sheets collection of the spreadbuilder object to the coordinates of the cell you need. The Cell object exposes 24 properties:

- Alignment
- BorderBottomColor
- BorderBottomStyle
- BorderDiagonalColor
- BorderDiagonalEnum
- BorderDiagonalStyle
- BorderLeftColor
- BorderLeftStyle
- BorderRightColor
- BorderRightStyle
- BorderTopColor
- BorderTopStyle
- FillColor
- FontBold
- FontItalic
- FontName
- FontSize
- FontUnderlineStyle
- ForeColor
- NumberFormat
- TextAngle
- Type
- Value
- VertAlignment

Note: *See DDCell Properties for full documentation.*

When using Cell properties to build a sheet, the FontSize property sets the Row Height property to the height of the font when the font height is greater than the row height.

Example:

```
sb.Sheets(0).Cell(0,0).Value = "Name"
sb.Sheets(0).Cell(0,0).FontBold = True
```

Designating Page Breaks

To create a pagebreak in an exported Excel document, use the AddHorizontalPageBreak method or the AddVertPageBreak method.

1. Access the sheet in which you need to create pagebreaks through the Sheets Collection in the spreadbuilder object.
2. Call AddHorizontalPageBreak to insert a break for a report that runs horizontally across the page, or call AddVertPage break to add a break in a report that runs vertically down the page.
3. The AddHorizontalPageBreak method has three arguments. The last two arguments are written to BIFF8 (Excel 97) files only, so in other cases these values may be ignored.
    - The first argument, *Row as Integer* is the only critical argument. Since this number is zero based, to break before Excel row 25, pass the value 24.
    - The next two zero based numbers define the starting and ending columns for the pagebreak.
    - If you don't want to define different pagebreaks at different columns, use AddHorizonalPageBreak (xxx,0,255) to specify all columns.
4. The AddVerticalPageBreak method is complementary to the AddHorizontalPageBreak method.
    - The first argument designates the column for the break.
    - The second and third arguments are the range of rows for the vertical pagebreak.
    - To specify all rows, use AddVerticalPageBreak (xxx,0,65535).
5. View the results in Excel by selecting PageBreak Preview under the view menu.

Example:

```
Private Sub MeExpt_Click()
   Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).AddHorizontalPageBreak (10,0,255)
   sb.Save "C:\AddressBook.xls"
End Sub
```

Adding Pictures

Use the AddPicture method to add a picture to your sheet. The AddPicture method takes 15 parameters:

```
AddPicture(pic As StdPicture,
   colL As Integer,
   dxL As Integer,
   rwT As Integer,
   dyT As Integer,
   colR As Integer,
   dxR As Integer,
   rwB As Integer,
   dyB As Integer,
   [BorderLineWeight],
   [BorderLineStyle],
   [BorderColor As OLE_COLOR],
   [FillBackground],
   [BackColor As OLE_COLOR],
   [MoveType As SBFloatingMoveType])
```

Note: *See the AddPicture method in the Object Model for definitions of the parameters.*

Example:

```
Dim spread As New SpreadBuilder
Dim sheet As DDSheet
Set sheet = spread.Sheets(0)

'GetPic is an object typed function.
'The critical line calls LoadPicture
'Set GetPic = LoadPicture("MyPictureName")

Dim a, b
b = countPics()

For a = 1 To b
    sheet.Cell(a * 5, 0) = "pic " & a
'output our list of pictures at every 5th cell
'put the left edge at column c (2) , right edge at column f (5)
'the picture will be 4 cells tall, 3 cells wide
'will have solid red borders, with a weight of 2.
'we are using this with a transparent gif,
'so a blue background will be visible
'the image will move but not re-size when cells bounds
'are moved in excel
    sheet.AddPicture GetPic(a), 2, 0, a * 5, 0, 5, 0, a * 5 + 4,     0, 2, 0,
```

```
vbRed, 1, vbBlue, SBMoveNoSize
Next a

sheet.Name = "Version " & spread.Version
spread.Save "c:\test.xls"
```

Adding Lines

Use the AddLine method to add a line to your sheet. The AddLine method takes 10 parameters:

```
sb.Sheets(0).AddLine(
   colL As Integer,
   dxL As Integer,
   rwT As Integer,
   dyT As Integer,
   colR As Integer,
   dxR As Integer,
   rwB As Integer,
   dyB As Integer,
   iQu As SBLineDirection,
   [Weight As Long],
   [Style As Long],
   [Color As OLE_COLOR],
   [fAuto As Boolean],
   [MoveType As SBFloatingMoveType],
   [StartAHStyle As Long],
   [StartAHWidth As Long],
   [StartAHLength As Long],
   [EndAHStyle As Long],
   [EndAHWidth As Long],
   [EndAHLength As Long])
```

Note: *See the AddLine method in the Object Model for definitions of the parameters.*

TroubleShooting The Excel Export Filter and SpreadBuilder

Using the Cell to build a sheet but all values are converted to zero.

1. First determine what the cell Type is when you set the number.
   Use
   ```
   ? sheet.Cell(0,0).type
   ```
   or
   ```
   ? cell.type
   ```
   to retrieve the enumerated type from the immediate window.
2. The default cell Type is SBBlank, which promotes only to SBLabel. This means that if you set a cell to a certain type and then change the type to Number, Spreadbuilder will not attempt to convert the number String to Number format. Rather, the Number cell is initialized to zero.

Lines are not aligning

Using lines to delimit columns can result in improper looking Excel files. Using borders for the edges of elements will solve the problem, as borders will be placed on columns in the same manner that data fields are. Lines are better used to bound pictures, because Excel floats both lines and pictures above the sheet.

Lines don't align with other lines.

This is an unfortunate result of how the export tries to preserve text at the expense of correct positioning when using the TrimEmptySpace property. This problem could be resolved by changing TrimEmptySpace to False, by using borders rather than lines, and by changing MinRowHeight. If these methods are not an option, try moving the Report elements so that the lines do not cross a line boundary of overlapping elements.

Rectangles, Rounded Rectangles and Ellipses aren't showing up.

The Excel Export Filter does not support these shapes.

Row size will not increase when TrimEmptySpace is True.

When TrimEmptySpace is True, the height of the row is always the height of the largest control on the row plus the value of the BorderSpace property. This may default to 59 twips if the largest element on a line is a textbox or label.

Numbers appear as "XXXXXX"

This occurs when there is not enough space in a column to display the number as formatted and is a result of the export filter incorrectly sizing the column.

To increase the size of the column, which is too small for the number field, try any of the following:

- Set DoubleBoundaries to True if there is no left aligned control overlapping the number field.
- Increase MinColumnWidth.
- Reposition elements in the report so that no elements overlap the number field's boundaries.
- Place unseen elements such as empty textboxes with a white border or background to give the export "hints" on where to place column boundaries. You must set a custom border, as empty textboxes without backgrounds or borders are removed by the export. The export is sensitive to the number of "hints."

ACTIVEREPORTS RTF EXPORT FILTER

What is the ActiveReports RTF Export Filter?

The *ActiveReports RTF Export Filter* is a dll-based library that exports an *ActiveReports* report to Rich Text Format.

Registering the RTF Export Library

The *ActiveReports RTF Export Filter* must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

1. Place the *rtfexpt.dll* file on your machine where you want it in your directory structure.
2. Open the command line by choosing **Run** from the **Start** menu.
3. Type the following in the **Open** box where [*filepath*] represents the path through the directory structure on your machine to where you have placed the .dll library:

```
regsvr32 [filepath]/rtfexpt.dll
```

4. A message box indicates that the registration has succeeded.

ACTIVEREPORTS PDF EXPORT FILTER

What is the ActiveReports PDF Export Filter?

The *ActiveReports PDF Export Filter* is a dll-based library that exports an *ActiveReports* report to Portable Document Format.

Registering the PDF Export Library

The *ActiveReports PDF Export Filter* must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

1. Place the *pdfexpt.dll* file on your machine where you want it in your directory structure.
2. Open the command line by choosing **Run** from the **Start** menu.
3. Type the following in the **Open** box where [*filepath*] represents the path through the directory structure on your machine to where you have placed the .dll library:

```
regsvr32 [filepath]/pdfexpt.dll
```

4. A message box indicates that the registration has succeeded.

ACTIVEREPORTS TEXT EXPORT FILTER

What is the ActiveReports Text Export Filter?

The *ActiveReports Text Export Filter* is a dll-based library that exports an *ActiveReports* report to a text file. This export supports both Unicode and ANSI formats and can be used to export all text in the report into a delimited data file or an *approximated* text layout.

Registering the Text Export Library

The *ActiveReports Text Export Filter* must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

1. Place the *textexpt.dll* file on your machine where you want it in your directory structure.
2. Open the command line by choosing **Run** from the **Start** menu.
3. Type the following in the **Open** box where [*filepath*] represents the path through the directory structure on your machine to where you have placed the .dll library:

```
regsvr32 [filepath]/textexpt.dll
```

4. A message box indicates that the registration has succeeded.

ACTIVEREPORTS TIFF EXPORT FILTER

What is the ActiveReports TIFF Export Filter?

The *ActiveReports TIFF Export Filter* is a dll-based library that exports an *ActiveReports* report to TIFF file. This can be used to send reports as faxes.

Registering the TIFF Export Library

The *ActiveReports TIFF Export Filter* must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

1. Place the *tiffexpt.dll* file on your machine where you want it in your directory structure.
2. Open the command line by choosing **Run** from the **Start** menu.
3. Type the following in the **Open** box where [*filepath*] represents the path through the directory structure on your machine to where you have placed the .dll library:

```
regsvr32 [filepath]/tiffexpt.dll
```

4. A message box indicates that the registration has succeeded.

ACTIVEREPORTS' HTML EXPORT FILTER

What is the ActiveReports' HTML Export Filter

The *ActiveReports HTMLExport Filter* is a dll-based library that exports an *ActiveReports* report to HTML format.

Note: *Lines are not exported to HTML*

Registering the HTML Export Library

The *ActiveReports HTML Export Filter* must be registered on your machine before you can use the filter in a Visual Basic project.

You can use the command line to register the .dll library via the following steps:

5. Place the *htmlexpt.dll* file on your machine where you want it in your directory structure.
6. Open the command line by choosing **Run** from the **Start** menu.
7. Type the following in the **Open** box where [*filepath*] represents the path through the directory structure on your machine to where you have placed the .dll library:

```
regsvr32 [filepath]/htmlexpt.dll
```

8. A message box indicates that the registration has succeeded.

Common Properties

Property	Data Type	Description
FileName	String	Specifies the name of the file to which the report will export.

FileName

Description

Specifies the name of the file to which the report will export.

Syntax

ExcelExportObject.FileName [= Value]

Data Type

String

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

Common Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportStream	Exports a report's pages collection to a byte array. **Sub** ExportStream(*ActiveReportsPagesCollection* as Unknown, *OutputParamter*)

Export

Discription

Exports a pages collection to the specified document format. The file is determined by the Filename property. The Export method requires a pages collection, which is created after a report, is run. You can call the Export method directly from the report object or from the viewer's pages collection.

Return Type

None

Syntax

```
Sub Export(pagesColl as Unknown)
```

Parameters

Parameters

Name	Type	Description
pagesColl	Unknown	The report’s pages collection.

Example

```
Private Sub mEExpt_Click()
   Dim xls As New ActiveReportsExcelExport.ARExportExcel
   Dim sFile As String
   Dim bSave As Boolean
   sFile = "c:\activereports\vbexport\test1.xls"
   xls.FileName = sFile
   If arv.Pages.Count > 0 Then
      xls.Export arv.Pages
   End If
End Sub
```

ExportStream

Description

Exports the report's pages collection to a byte array.

Return Type

None

Syntax

```
Sub ExportStream(pagesColl as Unknown, OutputParameter)
```

Parameters

Name	Type	Description
pagesColl	Variant	A report's pages collection.
OutputParamter	Variant	Variant or byte variable to use.

Example

```
Private Function WebCacheExport() As Long
Dim myBArray As Variant
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

    myPDFExport.AcrobatVersion = DDACR40
    myPDFExport.FileName = App.Path & "\PDFReport.PDF"
    myPDFExport.JPGQuality = 100
    myPDFExport.OutputTOCAsBookmarks = True
    myPDFExport.SemiDelimitedNeverEmbedFonts = ""
    myPDFExport.ShowBookmarksInAcrobat = True

    myPDFExport.ExportStream rptInvoice.Page, myBArray
End Function
```

Common Events

Event	Description
OnProgress	This event allows the export filter's progress to be tracked. **Sub** OnProgress(ByVal PageNumber As Long)

OnProgress

Description

This event allows the export filter's progress to be tracked. In order to gain access to this event in each export filter "WithEvents" must be used when dimming the variable.

Syntax

```
Sub OnProgress(ByVal PageNumber as Long)
```

Parameters

Name	Description
PageNumber	The page the export filter is currently exporting

Example

```
Dim WithEvents myExportObject As ActiveReportExcelExport.ARExportExcel

Private sub myExportFilter_OnProgress(ByVal PageNumber As Long)
   FrmMain.lblexpcnt.caption = = " Exporting Page " & PageNumber & " of " &
rpt.Pages.Count
End Sub
```

Excel Export Properties

Property	Data Type	Description
AutoRowHeight	Boolean	When set to true, Excel will correct for the size of the line by resizing the line to the largest object on the line.
BorderSpace	Integer	Defines in twips the height added to a cell to prevent vertical cell borders from overlapping characters.
DoubleBoundaries	Boolean	Use the double boundaries property when text strings force the export to place columns on both the left and right sides of a field.
FileName	String	Specifies the name of the file to which the report will export.
GenPageBreaks	Boolean	Determines if the Export method will generate page breaks automatically in the exported file.
MinColumnWidth	Long	Specifies in twips how small columns can be in the Excel document. Larger values reduce number of columns in a sheet.
MinRowHeight	Long	Specifies in twips how small rows can be in the exported file. Larger values force the export to place more controls on a single line.
Multisheet	Boolean	Determines if the report will be generated as a single Excel sheet, or as a multiple sheet workbook.
ShowMarginSpace	Boolean	Specifies whether the space between the report elements and the margin will display. Default is False.
TrimEmptySpace	Boolean	Determines if the exported report outputs runs of vertical empty spaces, or if they are eliminated.
Version	Integer	Sets the Excel version of the exported file.

AutoRowHeight

Description

When set to true, Excel will correct for the size of the line by resizing the line to the largest object on the line.

Syntax

```
ExcelExportObject.AutoRowHeight [= Value]
```

Data Type

Boolean

Settings

Value	Description
True	Row heights for all sheets are automatically corrected by Excel.
False	Row heights Excel will not correct row heights calculated by the export.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

BorderSpace Property

Description

Defines in twips the height added to a cell to prevent vertical cell borders from overlapping characters. If set to zero, you will get more accurate placement of fields, but you may experience ugly output as Excel draws bottom borders over character descenders. That is, Excel may visually clip the bottom of 'g', 'j', or 'q' characters. The default value will work for most reports.

Syntax

ExcelExportObject.BorderSpace [= Value]

Data Type

Integer (in twips) Default = 59 twips.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

DoubleBoundaries Property

Description

Use the double boundaries property when text strings force the export to place columns on both the left and right sides of a field. This issue doesn't occur in most reports.

Syntax

```
ExcelExportObject.DoubleBoundaries [= Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Right aligned elements replace left aligned elements in the same column.
False	Right aligned elements do not replace left aligned elements.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

GenPageBreaks

Description

Determines if the Export method will generate page breaks automatically in the exported file.

Syntax

```
ExcelExportObject.DoubleBoundaries [= Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Export automatically places horizontal page breaks below the bottom-most element on each report page.
False	Will not create automatic page breaks.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

MinColumnWidth

Description

Specifies in twips how small columns can be in the Excel document. Larger values reduce number of columns in a sheet.

Note: *This is typically the single important variable to tweak when exporting a pages collection.*

Syntax

ExcelExportObject.MinColumnWidth [= Value]

Data Type

Long (in twips)

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

MinRowHeight

Description

Specifies in twips how small rows can be in the exported file. Larger values force the export to place more controls on a single line.

Syntax

ExcelExportObject.MinRowHeight [= Value]

Data Type

Long (in twips) Default = 128 twips.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

MultiSheet

Description

Determines if the report will be generated as a single Excel sheet, or as a multiple sheet workbook. Each page in the report will be placed on its own Excel sheet.

Syntax

ExcelExportObject.MultiSheet [= Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Each report page will result in a separate sheet in the Excel workbook.
False	Pages in the report will be formatted to the same Excel sheet. The first page in the report will start at row 1, and subsequent pages will be placed below.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

ShowMarginSpace

Description

Specifies whether the space between the report elements and the margin will display. Default is False. If set to True, the space between elements and the margin will be shown when TrimEmptySpace is False.

Syntax

ExcelExportObject.ShowMarginSpace [= Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	The space between the report elements and the margin displays if the TrimEmptySpace property is False.
False	The space between the report elements and the margin will not display.

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

TrimEmptySpace

Description

Determines if the exported report outputs runs of vertical empty spaces, or if they are eliminated.

Syntax

```
ExcelExportObject.TrimEmptySpace [= Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Vertical empty spaces are removed from the exported report.
False	Vertical empty spaces remain in the exported report

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

Version

Description

Sets the Excel version of the exported file. Versions 2,3,4,5,7 and 8 are supported.

Syntax

```
ExcelExportObject.Version [= Value]
```

Data Type

Integer (Default = 8)

Example

```
Private Sub RunReport()
   Dim xls as New ARExportExcel
   Dim rpt As New ActiveReport1

   xls.AutoRowHeight = True
   xls.BorderSpace = 0
   xls.DoubleBoundaries = True
   xls.FileName = "C:\Report1.XLS"
   xls.GenPageBreaks = True
   xls.minColumnwidth = 2000
   xls.MinRowheight = 26
   xls.Multisheet = False
   xls.ShowMarginSpace = True
   xls.TrimEmptySpace = True

   xls.Version = 8
   rpt.Run False
   xls.Export rpt.Pages

   Set rpt = Nothing
   Set xls = Nothing
End Sub
```

Excel Export Methods

Method	Description
Export	Exports a pages collection to Excel document determined by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportWebCache	Exports a pages collection to an Excel document stored in the WebCache. **Function** ExportWebCache(pagesColl as Unknown) As Long

ExportWebCache

Description

Exports a pages collection to an Excel document stored as a byte array. The method returns the ID for the new item added to the webcache. The ExportWebCache method requires a pages collection, which is created after a report, is run. You can call the ExportWebCache method directly from the report object or from the viewer's pages collection.

Return Type

Long

Syntax

```
Function ExportWebCache(pagesColl as Unknown) As Long
```

Parameters

Parameters

Name	Type	Description
pagesColl	Unknown	The report’s pages collection.

Example

```
Function WebCacheExport() As Long
    Dim myExportObject As ActiveReportsExcelExport.ARExportExcel
    Set myExportojbect = New ActiveReportsExcelExport.ARExportExcel

    WebCacheExport = myExportObject.ExportWebCache(rptInvoice.pages)
End Function
```

SpreadBuilder API Constants

SBCellType

Value	Mnemonic	Description
0	SBBlank	Blank Cell
1	SBLabel	Label Cell
2	SBNumber	Number Cell
3	SBBoolean	Boolean Cell

SBFloatingMoveType Constant

Value	Mnemonic	Description
0	SBMoveNoSize	Lines and Pictures move but do not size with cells.
1	SBMoveSize	Lines and Pictures move and size with cells.
2	SBNoMoveNoSize	Lines and Pictures do not move or size with cells.

SBHorizAlign Constant

Value	Mnemonic	Description
0	SBAlignGeneral	No specific alignment setting.
1	SBAlignLeft	Align left.
2	SBAlignCenter	Align center.
3	SBAlignRight	Align right.
4	SBAlignFill	Fill the space.
5	SBAlignJustify	Justify.
6	SBAlignCenterAcrossSelection	Center across selection

SBLineDirection Constant

Value	Mnemonic	Description
0	LowerRightToUpperLeft	
1	LowerLeftToUpperRight	
2	UpperLeftToLowerRight	
3	UpperRightToLowerLeft	

SBUnderlineStyle Constant

Value	Mnemonic	Description
0	SBNone	
1	SBSingle	
2	SBDouble	
22	SBDoubleAcc	
33	SBSingleAcc	

SBVertAlignment Constant

Value	Mnemonic	Description
0	SBVertAlignTop	Align top.
1	SBVertAlignCenter	Align center.
2	SBVertAlignBottom	Align bottom.
3	SBVertJustify	Justify.

SpreadBuilder API Properties

Property	Data Type	Description
ProtectWorkbookStructure	Boolean	Sets an Excel option to protect workbook structure.
ProtectWorkbookWindows	Boolean	Sets an Excel option to protect the document window from being moved.
Sheets	Sheets collection	Accesses the sheets collection. The property is read only.
Version	Integer	Sets the Excel version of the exported file.

ProtectWorkbookStructure

Description

Sets an Excel option to protect workbook structure. If this property is set to true, Excel will not let the user add, delete, reorder, or rename sheets in the workbook.

Syntax

SpreadBuilder.ProtectWorkbookStructure [= Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Workbook structure is protected.
False	Workbook structure is not protected.

Example

```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Save App.Path & "\AddressBook.xls"
   set sb = Nothing
End Sub
```

ProtectWorkbookWindows

Description

Sets an Excel option to protect the document window from being moved. If this property is set to true, Excel will not let the user directly move, resize, make hidden, make unhidden, or close the document window.

Syntax

```
SpreadBuilder.ProtectWorkbookWindows [= Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Workbook structure is protected.
False	Workbook structure is not protected.

Example

```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Save App.Path & "\AddressBook.xls"
   set sb = Nothing
End Sub
```

Sheets

Description

Accesses the sheets collection. The property is read only.

Syntax

SpreadBuilder.Sheets

Data Type

Sheets collection

Example

```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Save App.Path & "\AddressBook.xls"
   set sb = Nothing
End Sub
```

Version

Description

Sets the Excel version for SpreadBuilder to use. Versions 2,3,4,5,7 and 8 are supported.

Syntax

```
SpreadBuilder.Version [= Value]
```

Data Type

Integer (Default = 8)

Example

```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Version = 8
   sb.Save App.Path & "\AddressBook.xls"
   set sb = Nothing
End Sub
```

SpreadBuilder API Methods

Method	Description
Clear	The Clear method deletes all data and sheets created, resetting the Spreadbuilder object to its default state. **Sub** Clear()
GetSaveCaps	The return value is set with a bitmapped description of potential errors for saving the sheet. **Function** GetSaveCaps() As Long
Save	Saves spreadsheet content to an Excel document. **Sub** Save(FileName As String)

Clear

Description

The Clear method deletes all data and sheets created, resetting the Spreadbuilder object to its default state.

Syntax

```
Sub Clear()
```

Parameters

None

Example

```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Save App.Path & "\AddressBook.xls"
   sb.Clear
   set sb = Nothing
End Sub
```

GetSaveCaps

Description

The return value is set with a bitmapped description of potential errors for saving the sheet. Because all Spreadbuilder features are not compatible with all versions of Excel documents, the GetSaveCaps method allows you to test for an error or unsupported condition before saving. It is not necessary to call GetSaveCaps. Sheets can always be saved in any version, although styles and properties may be mapped to the best available in the current version.

Return Type

Long

Return Settings

bit 0 - Specifies if errors are version related.

bit 1 - Specifies the number of sheets that can be saved properly.

bit 2 - Specifies if the XF records can be saved properly.

bit 3 - Specifies if the font records can be saved properly.
bit 4 -Specifies if the number of colors will be decreased in writing the file.

Syntax

```
Function GetSaveCaps() As Long
```

Parameters

None

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
Dim lSaveCaps As Long
Dim messageboxresult
messageboxresult = vbOK
Dim sMsg As String
   sb.Sheets(0).Name = "AddressBook"
   With sb.Sheets(0)
      ' Headers
      .Cell(0,0).Value = "Name"
      .Cell(0,0).FontBold = True
      .Cell(0,1).Value = "Telephone"
      .Cell(0,1).FontBold = True
      ' Record 1
      .Cell(1,0).Value = "Joy Rosen"
      .Cell (1,1).Value = "(212) 890-9876"
   End With
   lSaveCaps = sb.GetSaveCaps()
   If lSaveCaps <> 0 Then
      sMsg = "Unable to save file: " & vbCrLf
      sMsg = sMsg & IIf((lSaveCaps Or &H1) = &H1, _
```

```
               “You may lose style and formatting information if you save in
this version" & vbCrLF, _
               "")
      sMsg = sMsg & IIf((lSaveCaps Or &H2) = &H2, _
               "Can’t save all sheets" & vbCrLF, _
               "")
      sMsg = sMsg & IIf((lSaveCaps Or &H4) = &H4, _
               "Can’t save all cell styles" & vbCrLF, _
               "")
      sMsg = sMsg & IIf((lSaveCaps Or &H8) = &H8, _
               "Can’t save all font styles" & vbCrLF, _
               "")
      sMsg = sMsg & IIf((lSaveCaps Or &H10) = &H10, _
               "Number of colors will be decreased" & vbCrLF, _
               "")

   SMsg = sMsg & “Would you like to save anyway?”
   messageboxresult = MsgBox (sMsg, vbOKCancel)

   End If

   If vbOk = messageboxresult Then sb.Save "C:\AddressBook.xls"

End Sub
```

Save

Description
Saves spreadsheet content to an Excel document.

Return Type

Long

Syntax
```
Sub Save(FileName As String)
```

Parameters

Name	Type	Description
FileName	String	Path and filename for the file to be saved as.

Example
```
Private Sub CreateExcelFile()
Dim sb As New SpreadBuilder
   sb.ProtectWorkbookStructure = False
   sb.ProtectWorkbookWindows = False
   sb.Sheets(0).Name = "Address Book"

   ' Headers
   sb.Sheets(0).Cell(0,0).Value = "Name"
   sb.Sheets(0).Cell(0,0).FontBold = True
   sb.Sheets(0).Cell(0,1).Value = "Telephone"
   sb.Sheets(0).Cell(0,1).FontBold = True
   sb.Sheets(0).Cell(0,1).FontItalic = True

   ' Record 1
   sb.Sheets(0).Cell(1,0).Value = "Joy Rosen"
   sb.Sheets(0).Cell (1,1).Value = "(212) 890-9876"

   sb.Save App.Path & "\AddressBook.xls"
   set sb = Nothing
End Sub
```

SpreadBuilder DDCell Properties

Property	Data Type	Description
Alignment	SBHorzAlignment	Sets alignment within the cell.
BorderBottomColor	OLE_Color	Sets the color of the bottom border of the cell.
BorderBottomStyle	Integer	Sets the style of the bottom border of the cell.
BorderDiagonalColor	OLE_Color	Sets the color of the diagonal of the cell.
BorderDiagonalEnum	Integer	Describes the existence of either diagonal through an enumerated value.
BorderDiagonalStyle	Integer	Sets the style of both cell diagonals.
BorderLeftColor	OLE_Color	Sets the color of the left border of the cell.
BorderLeftStyle	Integer	Sets the style of the left border of the cell.
BorderRightColor	OLE_Color	Sets the color of the right border of the cell.
BorderRightStyle	Integer	Sets the style of the left border of the cell.
BorderTopColor	OLE_Color	Sets the color of the top border of the cell.
BorderTopStyle	Integer	Sets the color of the top border of the cell.
ForeColor	OLE_Color	Sets the text color of the cell.
FillColor	OLE_Color	Sets the background fill color of the cell.
ForeBold	Bolean	Specifies whether or not the font appears bold.
FontItalic	Boolean	Specifies whether or not the font appears italicized.
FontName	String	Specifies the name of the font to use.
FontSize	Integer	Specifies the size of the font in ten-thousandth of a point.
FontUnderlineStyle	SBUnderlineStyle	Specifies the style of text underlining.
Hyperlink	String	Specifies a hyperlink.
NumberFormat	String	Sets the format of the numbers.
TextAngle	Integer	Angle in degrees of the Cell's contents.
Type	SBCellType	Sets the data type for the cell.
Value	Variant	Specifies the data contained within the cell.
VertAlignment	SBVertAlignment	Sets the vertical alignment within the cell.

Alignment

Description

Sets alignment within the cell. Uses SBHorzAlignment constants.

Syntax

```
SpreadBuilder.Sheets(index).DDCell.Alignment=[Value]
```

Data Type

SBHorzAlignment

Settings

Value	Mnemonic	Description
0	SBAlignGeneral	No specific alignment setting.
1	SBAlignLeft	Align left.
2	SBAlignCenter	Align center.
3	SBAlignRight	Align right.
4	SBAlignFill	Fill the space.
5	SBAlignJustify	Justify.
6	SBAlignCenterAcrossSelection	Center across selection

Example

```
' Prepare sb As a spreadbuilder object
' ..
With sb.Sheets(0)
   .Cell(0,0).Alignment = SBAlignLeft
   .Cell(0,0).VertAlignment = SBVertAlignTop
End with
```

BorderBottomColor, BorderTopColor, BorderLeftColor, BorderRightColor

Description

Sets the color of the bottom, top, left or right border of the cell.

Syntax

*SpreadBuilder.*Sheets(index).DDCell.BorderBottomColor=[Value]

Data Type

OLE_Color

Settings

None

Example

```
' prepare sb As a spread builder object
' ..
With sb.Sheets(0)
   .Cell(0,0).Alignment = SBAlignLeft
   .Cell(0,0).BorderBottomColor = vbBlack
   .Cell(0,0).BorderBottomStyle = 2
   .Cell(0,0).BorderDiagonalColor = vbBlack
   .Cell(0,0).BorderDiagonalEnum = 1
   .Cell(0,0).BorderDiagonalStyle = 2
End with
```

BorderBottomStyle, BorderTopStyle, BorderLeftStyle, BorderRightStyle, BorderDiagonalStyle

Description

Sets the style of the bottom, top, left or right border of the cell.

Syntax

*SpreadBuilder.*Sheets(index).DDCell.BorderBottomStyle=[Value]

Data Type

Integer

Settings

Effect for Microsoft Excel 97 file format. (BIFF8)

0 None
1 Single width solid
2 Double width solid
3 Single width, closely spaced dashes.
4 Single width, small closely spaced dashes
5 Triple width solid
6 Single width solid double line
7 Single width dotted
8 Double width dashed
9 Single width dash dot
10 Double width dash dot
11 Single width dash dot dot
12 Double width dash dot dot
13 Double width slanted dash dot

Effect for versions 3, 4, 5, and 7. Version 2 does not support lines.

0 None
1 Single width solid
2 Double width solid
3 Single width, closely spaced dashes.
4 Single width, small closely spaced dashes
5 Triple width solid
6 Single width solid double line
7 Single width dotted

Example

```
' prepare sb As a spread builder object
' ..
With sb.Sheets(0)
   .Cell(0,0).Alignment = SBAlignLeft
   .Cell(0,0).BorderBottomColor = vbBlack
   .Cell(0,0).BorderBottomStyle = 2
```

```
   .Cell(0,0).BorderDiagonalColor = vbBlack
   .Cell(0,0).BorderDiagonalEnum = 1
   .Cell(0,0).BorderDiagonalStyle = 2
End with
```

BorderDiagonalColor

Description

Sets the color of the diagonal of the cell.

Syntax

SpreadBuilder.Sheets(index).DDCell.BorderDiagonalColor=[Value]

Data Type

OLE_Color

Settings

None

Example

```
' prepare sb As a spread builder object
' ..
With sb.Sheets(0)
   .Cell(0,0).Alignment = SBAlignLeft
   .Cell(0,0).BorderBottomColor = vbBlack
   .Cell(0,0).BorderBottomStyle = 2
   .Cell(0,0).BorderDiagonalColor = vbBlack
   .Cell(0,0).BorderDiagonalEnum = 1
   .Cell(0,0).BorderDiagonalStyle = 2
End with
```

BorderDiagonalEnum

Description

Describes the existence of either diagonal through an enumerated value.

Syntax

SpreadBuilder.Sheets(index).DDCell.BorderDiagonalEnum=[Value]

Data Type

Integer

Settings

0 No diagonals.

1 Diagonal down.

2 Diagonal up.

3 Diagonal up and diagonal down.

Example

```
' prepare sb As a spread builder object
' ..
With sb.Sheets(0)
   .Cell(0,0).Alignment = SBAlignLeft
   .Cell(0,0).BorderBottomColor = vbBlack
   .Cell(0,0).BorderBottomStyle = 2
   .Cell(0,0).BorderDiagonalColor = vbBlack
   .Cell(0,0).BorderDiagonalEnum = 1
   .Cell(0,0).BorderDiagonalStyle = 2
End with
```

ForeColor

Description

Sets the text color of the cell.

Syntax

SpreadBuilder.Sheets(index).DDCell.ForeColor=[Value]

Data Type

OLE_Color

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).ForeColor = vbRed
```

FillColor

Description

Sets the background fill color of the cell.

Syntax

SpreadBuilder.Sheets(index).DDCell.FillColor=[Value]

Data Type

OLE_Color

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).FillColor = vbRed
```

FontBold

Description

Specifies whether or not the font appears bold.

Syntax

SpreadBuilder.Sheets(index).DDCell.FontBold=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Font appears bold.
False	Font does not appear bold.

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).FontBold = True
```

FontItalic

Description

Specifies whether or not the font appears italicized.

Syntax

```
SpreadBuilder.Sheets(index).DDCell.FontItalic=[Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Font appears italicized.
False	Font does not appear italicized.

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).FontItalic = True
```

FontName

Description

Specifies the name of the font to use.

Syntax

SpreadBuilder.Sheets(index).DDCell.FontName=[Value]

Data Type

String

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).Name = "Arial"
```

FontSize

Description

Specifies the size of the font in ten-thousandth of a point.

Syntax

SpreadBuilder.Sheets(index).DDCell.FontSize=[Value]

Data Type

Integer

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).FontSize = 10.00 * 10000
```

FontUnderlineStyle

Description

Specifies the style of text underlining using the SBUnderlineStyle constants.

Syntax

```
SpreadBuilder.Sheets(index).DDCell.FontUnderlineStyle=[Value]
```

Data Type

SBUnderlineStyle

Settings

Value	Mnemonic	Description
0	SBNone	
1	SBSingle	
2	SBDouble	
22	SBDoubleAcc	
33	SBSingleAcc	

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).FontUnderlineStyle = SBDoubleAcc
```

HyperLink

Description

Sets a hyperlink string.

Syntax

```
.Sheets(index).DDCell.Hyperlink =[Value]
```

Data Type

String

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).Hyperlink = "http://www.datadynamics.com"
```

NumberFormat

Description

Sets the format of the numbers.

Syntax

```
.Sheets(index).DDCell.NumberFormat =[Value]
```

Data Type

String

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).NumberFormat = "???,??0.00"
```

TextAngle

Description

Angle in degrees of the Cell's contents

Syntax

SpreadBuilder.Sheets(index).DDCell.TextAngle=[Value]

Data Type

Integer

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).TextAngle = 30
```

Type

Description

Sets the data type for the cell

Syntax

SpreadBuilder.Sheets(index).DDCell.Type=[Value]

Data Type

SBCellType

Settings

Value	Mnemonic	Description
0	SBBlank	Blank Cell
1	SBLabel	Label Cell
2	SBNumber	Number Cell
3	SBBoolean	Boolean Cell

Example

```
Private sub MeExpt_onClick()
   sb1.Sheets(0).Cell(0,0).Type = SBBoolean
End Sub
```

Value

Description

Specifies the data contained within the cell

Syntax

SpreadBuilder.Sheets(index).DDCell.Value=[Value]

Data Type

Variant

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Cell(0,0).Value = "John Smith"
```

VertAlignmentProperty

Description

Sets the vertical alignment within the cell.

Syntax

```
SpreadBuilder.Sheets(index).DDCell.VertAlignment=[Value]
```

Data Type

SBVertAlignment

Settings

Value	Mnemonic	Description
0	SBVertAlignTop	Align top.
1	SBVertAlignCenter	Align center.
2	SBVertAlignBottom	Align bottom.
3	SBVertJustify	Justify.

Example

```
' Prepare sb As a spreadbuilder object
' ..
With sb.Sheets(0)
   .cell(0,0).Alignment = SBAlignLeft
   .cell(0,0).VertAlignment = SBVertAlignTop
End with
```

SpreadBuilder DDColumn Properties

Property	Data Type	Description
Width	Long	Specifies the width of a column

Width

Description

Specifies the width of a column.

Syntax

SpreadBuilder.Sheets(index).DDColumn.Width=[Value]

Data Type

Long

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Columns(1).Width = 50
```

SpreadBuilder DDRow Properties

Property	Data Type	Description
AutoSize	Boolean	Specifies whether or not a row's height is auto sized to fit the contents.
Height	Integer	Specifies the height of the row.

AutoSize

Description

Specifies whether or not a row's height is auto sized to fit the content.

Syntax

SpreadBuilder.Sheets(index).DDRow.AutoSize=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	A row's height is auto sized to fit the content.
False	A row's height is not auto sized to fit the content.

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Rows(3).AutoSize = True
```

Height

Description

Specifies the height of the row.

Syntax

SpreadBuilder.Sheets(index).DDRow.Height=[Value]

Data Type

Integer (in twips) Default = 255.

Settings

None

Example

```
' Prepare sb As a spreadbuilder object
' ..
sb.Sheets(0).Rows(4).Height = 300
```

SpreadBuilder DDSheet Properties

Property	Data Type	Description
Name	String	Specifies the name of the sheet.

Name

Description

Specifies the name of the sheet.

Syntax

SpreadBuilder.Sheets(index).Name=[Value]

Data Type

String

Settings

None

Example

```
Private sub MeExpt_onClick()
   sb1.Sheets(0).Name = "Report 1"
End sub
```

SpreadBuilder DDSheet Methods

Method	Description
AddHorizontalPageBreak	Adds a horizontal page break. **Sub** AddHorizontalPageBreak(*row* **As Integer**, *columnstart* **As Integer**, *columnend* **As Integer**)
AddLine	Adds a horizontal page break. **Sub** AddLine(*colL* **As Integer**, *dxL* **As Integer**, *rwT* **As Integer**, *dyT* **As Integer**, *colR* **As Integer**, *dxR* **As Integer**, *rwB* **As Integer**, *dyB* **As Integer**, *rwB* **As Integer**, *dyB* **As Integer**, *iQu* **As SBLineDirection**, [*weight* **As Long**], [*Style* **As Long**], [*Color* **As OLE_Color**], [*fAuto* **As Boolean**], [*MoveType* **As SBFloatingMoveType**], [*StartAHStyle* **As Long**], [*StartAHWidth* **As Long**], [*StartAHLength* **As Long**], [*EndAHStyle* **As Long**], [*EndAHWidth* **As Long**], [*EndAHLength* **As Long**])
AddLinkPicture	Add a picture with hyperlink to the sheet. **Sub** AddLinkPicture(*pic* **As StdPicture**, *colL* **As Integer**, *dxL* **As Integer,** *rwT* **As Integer,** *dyT* **As Integer,** *colR* **As Integer,** *dxR* **As Integer,** *rwB* **As Integer,** *dyB* **As Integer,** *hyperlinkstring* **As String**, [*BorderLineWeight*], *[BorderLineStyle*],[*BorderColor* **As OLE_Color**], [*FillBackground*], [*BackColor* **As OLE_Color**], [*MoveType* **As SBFloatingMoveType**])
AddPicture	Adds a picture to a sheet. **Sub** AddLinkPicture(*pic* **As StdPicture**, *colL* **As Integer**, *dxL* **As Integer,** *rwT* **As Integer,** *dyT* **As Integer,** *colR* **As Integer,** *dxR* **As Integer,** *rwB* **As Integer,** *dyB* **As Integer,** [*BorderLineWeight*], *[BorderLineStyle*],[*BorderColor* **As OLE_Color**], [*FillBackground*], [*BackColor* **As OLE_Color**], [*MoveType* **As SBFloatingMoveType**])
AddVerticalPage	Adds a vertical page break. **Sub** AddVerticallPageBreak(*col* **As Integer**, *rowstart* **As Integer**, *rowend* **As Integer**)
Cell	Returns a cell object at the specified coordinates. **Sub** Cell(*row* **As Long**, *col* **As Long**)
Clear	Deletes all properties and content, creating a blank sheet. **Sub** Clear()
Columns	Returns a column object from the specified location. **Sub** Columns(*col* **As Long**)
Rows	Returns a row object from the specified location. **Sub** Rows(*row* **As Long**)

AddhorizontalPageBreak

Description
Adds a horizontal page break.

Return Type

None

Syntax

```
Sub AddHorizontalPageBreak(row As Integer, columnstart As Integer, columnend
As Integer)
```

Parameters

The AddHorizontalPageBreak method has three arguments. The last two arguments are written to BIFF8 (Excel 97 and later) files only, so in other cases these values may be ignored. The first argument, *Row as Integer* is the only critical argument.

- Since this number is zero based, to break before Excel row 25, pass the value 24.
- The next two zero based numbers define the starting and ending columns for the pagebreak.
- If you don't want to define different pagebreaks at different columns, use AddHorizonalPageBreak (xxx,0,255) to specify all columns.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
sb.Sheets(0).Name = "AddressBook" With sb.Sheets(0)
' Headers
With sb.Sheets(0)
         .Cell(0,0).Value = "Name"
         .Cell(0,0).FontBold = True
         .Cell(0,1).Value = "Telephone"
         .Cell(0,1).FontBold = True
         'Record 1
         .Cell(1,0).Value = "Joy Rosen"
         .Cell (1,1).Value = "(212) 890-9876"
End With
sb.Sheets(0).AddHorizontalPageBreak (10, 0, 255)
sb.Save "C:\AddressBook.xls"
sb.Clear
End Sub
```

AddLine

Description
Adds one line with specified coordinates and properties.

Return Type

None

Syntax

```
Sub AddLine(colL As Integer, dxL As Integer, rwT As Integer, dyT As Integer,
colR As Integer, dxR As Integer, rwB As Integer, dyB As Integer, rwB As
Integer, dyB As Integer, iQu As SBLineDirection, [weight As Long], [Style As
Long], [Color As OLE_Color], [fAuto As Boolean], [MoveType As
SBFloatingMoveType], [StartAHStyle As Long], [StartAHWidth As Long],
[StartAHLength As Long], [EndAHStyle As Long], [EndAHWidth As Long],
[EndAHLength As Long])
```

Parameters

Name	Type	Description
ColL	Integer	Left extent column.
rwT	Integer	Top extent row.
ColR	Integer	Right extent column
RwB	Integer	Bottom extent row

Note: *Passing zero will have the left edge of the line show in column 1 in excel.*

The remaining four required parameters are the endpoints of the line as cell height ratios and cell width ratios.

- **dxL** is 1024 times the ratio of the placement of the left extent of the line and the right border of the cell. Passing dxL as zero would correspond to the left edge of the cell, and 1023 would correspond to the far right. Passing 511 would put the left extent of the line roughly in the horizontal middle of the cell.
- **dxR** is 1024 times the ratio of the placement of the **right** extent of the line and the right border of the cell.
- **dyT** is 256 times the ratio of the placement of the top extent of the line and the bottom border of the cell. Passing dyT as zero would correspond to the top edge of the cell, and 255 would correspond to the very bottom. Passing 127 would put the top extent of the line roughly in the vertical middle of the cell.
- **dyB** is 256 times the ratio of the placement of the bottom extent of the line and the bottom border of the cell. Passing dyB as zero would correspond to the bottom edge of the cell.

iQu
Use SBLineDirection constants to define line direction.

0 LowerRight to UpperLeft

1 LowerLefttoUpperRight

2 UpperLefttoLowerRight

3 UpperRighttoLowerLeft

The optional parameters are defined as follows:

Weight
Set to one of the Excel enumerated line weight values.

Effect for Microsoft Excel 97 file format. (BIFF8)

Parameter will generate line with weight parameter as points.

Effect for versions 3, 4, 5, and 7. Version 2 does not support lines.

0 Hairline 0 points
1 Single .75 points
2 Double 1.35 points
3 Thick 1.95 points

Style
Set to one of the Excel enumerated line style values.

Effect for Microsoft Excel 97 file format. (BIFF8)

0 Solid
1 Small Dash (No Equivalent Menu for Excel 2000)
2 Square Dot
3 Small Dash, Square Dot (No Equivalent Menu for Excel 2000)
4 Small Dash, Square Dot, Square Dot (No Equivalent Menu for Excel 2000)
5 Loosely spaced Square Dot (No Equivalent Menu for Excel 2000)
6 Dash
7 Long Dash
8 Dash Dot
9 Long Dash Dot
10 Long Dash Dot Dot
11 Round Dot

Effect for versions 3, 4, 5, and 7. Version 2 does not support lines.

0 Solid
1 Long Dash
2 Dash
3 Long Dash Dot
4 Long Dash Dot Dot
5 Pattern 50% Solid (No Equivalent Menu for Excel 2000)
6 Pattern 75% Solid (No Equivalent Menu for Excel 2000)
7 Pattern 25% Solid (No Equivalent Menu for Excel 2000)
15 Transparent (No Line)

Color
OLE_Color

fAuto
Set fAuto to 1 if you want the Excel automatic border option turned on.

MoveType
Set to one of the SBFloatingMoveType constants

0 SBMoveNoSize Lines and Pictures move but do not size with cells.
1 SBMoveSize Lines and Pictures move and size with cells.
2 SBNoMoveNoSize Lines and Pictures do not move or size with cells.

AHStyle

Set ahstyle to the Excel enumerated type for arrowhead styles.

0 None
1 Open
2 Filled
3 Double-ended open
4 Double-ended filled

AHWidth
Set ahwidth to the Excel enumerated type for arrowhead width.

0 Narrow
1 Medium
2 Wide

AHLength
Set ahlength to the Excel enumerated type for arrowhead length.

0 Short
1 Medium
2 Long

Example

AddLinkPicture

Description
Adds a picture and hyperlink with specified coordinates and properties to a sheet object.

Return Type

None

Syntax

```
Sub AddLinkPicture(pic As StdPicture, colL As Integer, dxL As Integer, rwT As Integer, dyT As Integer, colR As Integer, dxR As Integer, rwB As Integer, dyB As Integer, hyperlinkstring As String, [BorderLineWeight], [BorderLineStyle], [BorderColor As OLE_Color], [FillBackground], [BackColor As OLE_Color], [MoveType As SBFloatingMoveType])
```

Parameters

Name	Type	Description
Pic	StdPicture	Path and filename of the image. LoadPicture("Logo.gif")
ColL	Integer	Left extent column.
rwT	Integer	Top extent row.
ColR	Integer	Right extent column
RwB	Integer	Bottom extent row
Hyperlinkstring	String	String hyperlink expression.

Note: *Passing zero will have the left edge of the line show in column 1 in excel.*

The remaining four required parameters are the endpoints of the line as cell height ratios and cell width ratios.

- **dxL** is 1024 times the ratio of the placement of the left extent of the line and the right border of the cell. Passing dxL as zero would correspond to the left edge of the cell, and 1023 would correspond to the far right. Passing 511 would put the left extent of the line roughly in the horizontal middle of the cell.
- **dxR** is 1024 times the ratio of the placement of the **right** extent of the line and the right border of the cell.
- **dyT** is 256 times the ratio of the placement of the top extent of the line and the bottom border of the cell. Passing dyT as zero would correspond to the top edge of the cell, and 255 would correspond to the very bottom. Passing 127 would put the top extent of the line roughly in the vertical middle of the cell.
- **dyB** is 256 times the ratio of the placement of the bottom extent of the line and the bottom border of the cell. Passing dyB as zero would correspond to the bottom edge of the cell.

BorderLineWeight

Set to one of the Excel enumerated line weight values.

Effect for Microsoft Excel 97 file format (BIFF8)

Set BorderLineWeight to the desired picture border line weight in points.

Effect for versions 3, 4, 5, and 7. Version 2 does not support Pictures.

0 Hairline 0 points

1 Single .75 points

2 Double 1.35 points

3 Thick 1.95 points

BorderLineStyle

Effect for Microsoft Excel 97 file format (BIFF8)

0 Solid
1 Small Dash (No Equivalent Menu for Excel 2000)
2 Square Dot
3 Small Dash, Square Dot (No Equivalent Menu for Excel 2000)
4 Small Dash, Square Dot, Square Dot (No Equivalent Menu for Excel 2000)
5 Loosely spaced Square Dot (No Equivalent Menu for Excel 2000)
6 Dash
7 Long Dash
8 Dash Dot
9 Long Dash Dot
10 Long Dash Dot Dot
11 Round Dot
12 Transparent (No Line)

Effect for versions 3, 4, 5 and 7. Version 2 does not support Pictures.

0 Solid
1 Long Dash
2 Dash
3 Long Dash Dot
4 Long Dash Dot Dot
5 Pattern 50% Solid (No Equivalent Menu for Excel 2000)
6 Pattern 75% Solid (No Equivalent Menu for Excel 2000)
7 Pattern 25% Solid (No Equivalent Menu for Excel 2000)
15 Transparent (No Line)

BorderColor

OLE_Color

Use the RGB function to generate an OLE_COLOR.

FillBackground

Set fls to True for a solid background fill pattern for your picture.

BackColor

OLE_Color

Use the RGB function to generate an OLE_COLOR.

MoveType

Set to one of the SBFloatingMoveType constants

0 SBMoveNoSize Lines and Pictures move but do not size with cells.
1 SBMoveSize Lines and Pictures move and size with cells.
2 SBNoMoveNoSize Lines and Pictures do not move or size with cells.

Example

```
Dim spread As New SpreadBuilder
```

```
Dim sheet As DDSheet
Set sheet = spread.Sheets(0)

‘GetPic is an object typed function.
‘The critical line calls LoadPicture
‘Set GetPic = LoadPicture(“MyPictureName”)

Dim a, b
b = countPics()

For a = 1 To b
    sheet.Cell(a * 5, 0) = "pic " & a
‘output our list of pictures at every 5th cell
‘put the left edge at column c (2) , right edge at column f (5)
‘the picture will be 4 cells tall, 3 cells wide
‘will have solid red borders, with a weight of 2.
‘we are using this with a transparent gif,
‘so a blue background will be visible
‘the image will move but not re-size when cells bounds
‘are moved in excel
    sheet.AddLinkPicture GetPic(a), 2, 0, a * 5, 0, 5, 0, a * 5 + 4,0,
"http://www.datadynamics.com", 2, 0, vbRed, 1, vbBlue, SBMoveNoSize
Next a

sheet.Name = "Version " & spread.Version
spread.Save "c:\test.xls"
```

AddPicture

Description
Adds a picture with specified coordinates and properties to a sheet object.

Return Type

None

Syntax

```
Sub AddPicture(pic As StdPicture, colL As Integer, dxL As Integer, rwT As
Integer, dyT As Integer, colR As Integer, dxR As Integer, rwB As Integer, dyB
As Integer, [BorderLineWeight], [BorderLineStyle],[BorderColor As OLE_Color],
[FillBackground], [BackColor As OLE_Color], [MoveType As SBFloatingMoveType])
```

Parameters

Name	Type	Description
Pic	StdPicture	Path and filename of the image. LoadPicture("Logo.gif")
ColL	Integer	Left extent column.
rwT	Integer	Top extent row.
ColR	Integer	Right extent column
RwB	Integer	Bottom extent row

Note: *Passing zero will have the left edge of the picture show in column 1 in excel.*

The remaining four required parameters are the endpoints of the line as cell height ratios and cell width ratios.

- **dxL** is 1024 times the ratio of the placement of the left extent of the line and the right border of the cell. Passing dxL as zero would correspond to the left edge of the cell, and 1023 would correspond to the far right. Passing 511 would put the left extent of the line roughly in the horizontal middle of the cell.
- **dxR** is 1024 times the ratio of the placement of the **right** extent of the line and the right border of the cell.
- **dyT** is 256 times the ratio of the placement of the top extent of the line and the bottom border of the cell. Passing dyT as zero would correspond to the top edge of the cell, and 255 would correspond to the very bottom. Passing 127 would put the top extent of the line roughly in the vertical middle of the cell.
- **dyB** is 256 times the ratio of the placement of the bottom extent of the line and the bottom border of the cell. Passing dyB as zero would correspond to the bottom edge of the cell.

BorderLineWeight

Set to one of the Excel enumerated line weight values.

Effect for Microsoft Excel 97 file format (BIFF8)

Set BorderLineWeight to the desired picture border line weight in points.

Effect for versions 3, 4, 5, and 7. Version 2 does not support Pictures.

0 Hairline 0 points

1 Single .75 points

2 Double 1.35 points

3 Thick 1.95 points

BorderLineStyle

Effect for Microsoft Excel 97 file format (BIFF8)

0 Solid
1 Small Dash (No Equivalent Menu for Excel 2000)
2 Square Dot
3 Small Dash, Square Dot (No Equivalent Menu for Excel 2000)
4 Small Dash, Square Dot, Square Dot (No Equivalent Menu for Excel 2000)
5 Loosely spaced Square Dot (No Equivalent Menu for Excel 2000)
6 Dash
7 Long Dash
8 Dash Dot
9 Long Dash Dot
10 Long Dash Dot Dot
11 Round Dot
12 Transparent (No Line)

Effect for versions 3, 4, 5 and 7. Version 2 does not support Pictures.

0 Solid
1 Long Dash
2 Dash
3 Long Dash Dot
4 Long Dash Dot Dot
5 Pattern 50% Solid (No Equivalent Menu for Excel 2000)
6 Pattern 75% Solid (No Equivalent Menu for Excel 2000)
7 Pattern 25% Solid (No Equivalent Menu for Excel 2000)
15 Transparent (No Line)

BorderColor

OLE_Color

Use the RGB function to generate an OLE_COLOR.

FillBackground

Set fls to True for a solid background fill pattern for your picture.

BackColor

OLE_Color

Use the RGB function to generate an OLE_COLOR.

MoveType

Set to one of the SBFloatingMoveType constants

0 SBMoveNoSize Lines and Pictures move but do not size with cells.
1 SBMoveSize Lines and Pictures move and size with cells.
2 SBNoMoveNoSize Lines and Pictures do not move or size with cells.

Example

```
Dim spread As New SpreadBuilder
Dim sheet As DDSheet
Set sheet = spread.Sheets(0)
```

```
‘GetPic is an object typed function.
‘The critical line calls LoadPicture
‘Set GetPic = LoadPicture(“MyPictureName”)

Dim a, b
b = countPics()

For a = 1 To b
    sheet.Cell(a * 5, 0) = "pic " & a
‘output our list of pictures at every 5th cell
‘put the left edge at column c (2) , right edge at column f (5)
‘the picture will be 4 cells tall, 3 cells wide
‘will have solid red borders, with a weight of 2.
‘we are using this with a transparent gif,
‘so a blue background will be visible
‘the image will move but not re-size when cells bounds
‘are moved in excel
    sheet.AddPicture GetPic(a), 2, 0, a * 5, 0, 5, 0, a * 5 + 4,0, 2, 0,
vbRed, 1, vbBlue, SBMoveNoSize
Next a

sheet.Name = "Version " & spread.Version
spread.Save "c:\test.xls"
```

AddVerticalPageBreak

Description
Adds a vertical pagebreak to the page at the specified location.

Return Type

None

Syntax

```
Sub AddVerticalPageBreak(row As Integer, rowstart As Integer, rowend As
Integer)
```

Parameters

The AddVerticalPageBreak method is complementary to the AddHorizontalPageBreak method.

- The first argument designates the column for the break.
- The second and third arguments are the range of rows for the vertical pagebreak.
- To specify all rows, use AddVerticalPageBreak (xxx,0,65535).

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook" With sb.Sheets(0)
' Headers
   With sb.Sheets(0)
      .Cell(0,0).Value = "Name"
      .Cell(0,0).FontBold = True
      .Cell(0,1).Value = "Telephone"
      .Cell(0,1).FontBold = True
      ' Record 1
      .Cell(1,0).Value = "Joy Rosen"
      .Cell (1,1).Value = "(212) 890-9876"
End With
   sb.Sheets(0).AddVerticalPageBreak (15, 0, 65535)
   sb.Save "C:\AddressBook.xls"
   sb.Clear
End Sub
```

Cell

Description
Returns a cell object at the specified coordinates.

Return Type

None

Syntax

```
Sub Cell(row As Long, column As Long)
```

Parameters

Name	Type	Description
Row	Long	The row number.
Column	Long	The column number.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook" With sb.Sheets(0)
'Headers
   With sb.Sheets(0)
         .Cell(0,0).Value = "Name"
         .Cell(0,0).FontBold = True
         .Cell(0,1).Value = "Telephone"
         .Cell(0,1).FontBold = True
         'Record 1
         .Cell(1,0).Value = "Joy Rosen"
         .Cell (1,1).Value = "(212) 890-9876"
   End With
   sb.Save "C:\AddressBook.xls"
End Sub
```

Clear

Description
Deletes all properties and content, creating a blank sheet.

Return Type

None

Syntax

```
Sub Clear()
```

Parameters
None

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook" With sb.Sheets(0)
' Headers
   With sb.Sheets(0)
         .Cell(0,0).Value = "Name"
         .Cell(0,0).FontBold = True
         .Cell(0,1).Value = "Telephone"
         .Cell(0,1).FontBold = True
         ' Record 1
         .Cell(1,0).Value = "Joy Rosen"
         .Cell (1,1).Value = "(212) 890-9876"
   End With
   sb.Sheets(0).AddHorizontalPageBreak (10, 0, 255)
   sb.Save "C:\AddressBook.xls"
   sb.Sheets(0).Clear
End Sub
```

Columns

Description
Returns a column object from the specified location.

Return Type

None

Syntax

```
Sub Columns(column As Long)
```

Parameters

Name	Type	Description
Column	Long	The column number.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Columns(0).Width = 255
End Sub
```

Rows

Description
Returns a row object from the specified location.

Return Type

None

Syntax

```
Sub Rows(Row As Long)
```

Parameters

Name	Type	Description
Row	Long	The row number.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Rows(0).Height = 255
End Sub
```

SpreadBuilder DDSheets Methods

Method	Description
Add	Adds a sheet object to the sheets collection. **Fucntion** Add(*Name* **As String,** [*insertAt*]) **As DDSheet**
Count	Returns a count of the sheet objects in the sheets collection. **Function** Count() **As Integer**
Item	Retrieves the sheet object indicated by the index. **Function** Item(*index*) **As DDSheet**
Move	Moves a sheet to a new location in the workbook. **Sub** Move(*Index*, *NewPostion* **As Integer**)
Remove	Removes a sheet from the sheets collections. **Sub** Remove(*index*)
Select	Sets the selected sheet. **Sub** Select(*index*)

Add

Description
Adds a sheet to the Sheets collection.

Return Type
DDSheet

Syntax
```
Function Add(Name As String, [insertAt]) As DDSheet
```

Parameters

Name	Type	Description
Name	String	Name for the sheet.

Example
```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets.Add = "AddressBook2"
End Sub
```

Count

Description

Returns an integer representing the total number of objects in the 0 based Sheets collection.

Return Type

Integer

Syntax

```
Function Count() As Integer
```

Parameters

None

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Column(0).Width = 255
   MsgBox sb.Sheets.Count
End Sub
```

Item

Description
Accesses a particular member of the Sheets collection by index number.

Return Type
DDSheet

Syntax
```
Function Item(index) As DDSheet
```

Parameters

Name	Type	Description
Index	Variant	Index number for the sheet.

Example
```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Column(0).Width = 255
   if isEmpty(sb.Sheets.Item(1)) then
      sb.Clear
   End if
End Sub
```

Move

Description
Repositions a particular member of the Sheets collection.

Return Type

None

Syntax

```
Sub Move(index, NewPosition As Integer)
```

Parameters

Name	Type	Description
Index	Variant	Index number for the sheet.
NewPostion	Integer	Position in the sheets collection to move the sheet to.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Column(0).Width = 255
   sb.Sheets.Move (3, 0)
End Sub
```

Remove

Description
Removes a specified object from the Sheets collection.

Return Type
None

Syntax
```
Sub Remove(index)
```

Parameters

Name	Type	Description
Index	Variant	Index number for the sheet.

Example
```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Column(0).Width = 255
   Sb.Sheets.Remove (0)
End Sub
```

Select

Description
Set a sheet in a Workbook to a selected state.

Return Type

None

Syntax

```
Sub Select(index)
```

Parameters

Name	Type	Description
Index	Variant	Index number for the sheet.

Example

```
Private Sub MeExpt_Click()
Dim sb as New ActiveReportsExcelExport.SpreadBuilder
   sb.Sheets(0).Name = "AddressBook"
   sb.Sheets(0).Column(0).Width = 255
   sb.Sheets.Select(0)
End Sub
```

RTF Export Properties

Property	Data Type	Description
FileName	String	Specifies the name of the file to which the report will export.

RTF Export Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportStream	Exports a report's pages collection to a byte array. Sub ExportStream(ActiveReportsPagesCollection as Unknown, OutputParamter)

PDF Export Properties

Property	Data Type	Description
AcrobatVersion	DDACRVersion	Controls which version of acrobat will load the export filter. If enumerated value DDACR21 (0) is use, compression will be disabled.
FileName	String	Specifies the name of the file to which the report will export.
JPGQuality	Long	Sets the quality of images exported to PDF. Range 1-100
OutputTOCAsBookMarks	Boolean	Determines whether or not the TOC pages will be included in the exported PDF file.
SemiDelimitedNeverEmbedFonts	String	List of fonts which will never be embedded into the PDF document.
ShowBookmarksInAcrobat	Boolean	Determines if Acrobat's Bookmarks Toolwindow is displayed when the PDF file is opened.

AcrobatVersion

Description

Sets the version of the Acrobat file exported.

Syntax

pdfExportObject.AcrobatVersion=[Value]

Data Type

DDACRVersion

Settings

Value	Mnemonic	Description
0	DDACR21	Adobe Acrobat Reader 2.1, without compression.
1	DDACR30	Adobe Acrobat Reader 3.0, with compression.
2	DDACR40	Adobe Acrobat Reader 4.0, with compression.

Example

```
Private Sub PDFExport()
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

   myPDFExport.AcrobatVersion = DDACR40
   myPDFExport.FileName = App.Path & "\PDFReport.PDF"
   myPDFExport.JPGQuality = 100
   myPDFExport.OutputTOCAsBookmarks = True
   myPDFExport.SemiDelimitedNeverEmbedFonts = ""
   myPDFExport.ShowBookmarksInAcrobat = True

   myPDFExport.Export rptInvoice.pages
End Sub
```

JPGQuality

Description

Sets the quality of images exported to PDF. Range: 0-100

Note: *the PDF Export will not export high-resolution images unless the image file is a metafile. In order to preserve the image's resolution, all high-resolution images should be converted to metafiles.*

Syntax

pdfExportObject.JPGQuality=[Value]

Data Type

Long

Settings

None

Example

```
Private Sub PDFExport()
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

   myPDFExport.AcrobatVersion = DDACR40
   myPDFExport.FileName = App.Path & "\PDFReport.PDF"
   myPDFExport.JPGQuality = 100
   myPDFExport.OutputTOCAsBookmarks = True
   myPDFExport.SemiDelimitedNeverEmbedFonts = ""
   myPDFExport.ShowBookmarksInAcrobat = True

   myPDFExport.Export rptInvoice.pages
End Sub
```

OutputTOCAsBookmarks

Description

Determines whether or not the report's TOC will be exported as PDF bookmarks. Setting the property to false will prevent the TOC items from being exported.

Syntax

pdfExportObject.OutputTOCAsBookmarks=[Value]

Data Type

Boolean (Default = True)

Settings

Value	Description
True	Export the TOC.
False	Do not export the TOC.

Example

```
Private Sub PDFExport()
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

   myPDFExport.AcrobatVersion = DDACR40
   myPDFExport.FileName = App.Path & "\PDFReport.PDF"
   myPDFExport.JPGQuality = 100
   myPDFExport.OutputTOCAsBookmarks = True
   myPDFExport.SemiDelimitedNeverEmbedFonts = ""
   myPDFExport.ShowBookmarksInAcrobat = True

   myPDFExport.Export rptInvoice.pages
End Sub
```

SemiDelimitedNeverEmbedFonts

Description

List of fonts that will never be embedded into the PDF document. Fonts are listed as a semicolon delimited string. Do not use spaces except between words of the font name. Use an empty string to export all fonts. A PDF containing embedded fonts have a significantly larger file size than a PDF which does not contain embedded fonts.

Syntax

pdfExportObject.SemiDelimitedNeverEmbedFonts=[Value]

Data Type

String

Default Setting

"Courier New;Times New Roman;Arial"

Example

```
Private Sub PDFExport()
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

   myPDFExport.AcrobatVersion = DDACR40
   myPDFExport.FileName = App.Path & "\PDFReport.PDF"
   myPDFExport.JPGQuality = 100
   myPDFExport.OutputTOCAsBookmarks = True
   myPDFExport.SemiDelimitedNeverEmbedFonts = ""
   myPDFExport.ShowBookmarksInAcrobat = True

   myPDFExport.Export rptInvoice.pages
End Sub
```

ShowBookMarksInAcrobat

Description

When the exported file is loaded into Acrobat, this property controls whether the Bookmarks tool window will either be displayed, or have default visibility.

Syntax

pdfExportObject.ShowBookmarksInAcrobat=[Value]

Data Type

Boolean (Default = True)

Settings

Value	Description
True	Shows the bookmarks tool window.
False	Does not show the bookmarks tool window.

Example

```
Private Sub PDFExport()
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

   myPDFExport.AcrobatVersion = DDACR40
   myPDFExport.FileName = App.Path & "\PDFReport.PDF"
   myPDFExport.JPGQuality = 100
   myPDFExport.OutputTOCAsBookmarks = True
   myPDFExport.SemiDelimitedNeverEmbedFonts = ""
   myPDFExport.ShowBookmarksInAcrobat = True

   myPDFExport.Export rptInvoice.pages
End Sub
```

PDF Export Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportStream	Exports a report's pages collection to a byte array. **Sub** ExportStream(*ActiveReportsPagesCollection* **As Unknown**, *OutputParamter*)
ExportWebCache	Exports the pages collection to the web cache service. **Function** ExportWebCache(*pagesCol*l **As Unknown**) **As Long**

ExportWebCache

Description
Exports the pages collection to the web cache service. When the pages are exported the web cache returns the ID for the item.

Return Type

Long

Syntax

```
Function ExportWebCache(pagesColl as Unknown) As Long
```

Parameters

Name	Type	Description
pagesColl	Variant	A report's pages collection.

Example

```
Private Function WebCacheExport() As Long
Dim myPDFExport As ActiveReportsPDFExport.ARExportPDF
Set myPDFExport = New ActiveReportsPDFExport.ARExportPDF

    myPDFExport.AcrobatVersion = DDACR40
    myPDFExport.FileName = App.Path & "\PDFReport.PDF"
    myPDFExport.JPGQuality = 100
    myPDFExport.OutputTOCAsBookmarks = True
    myPDFExport.SemiDelimitedNeverEmbedFonts = ""
    myPDFExport.ShowBookmarksInAcrobat = True

    myPDFExport.Export rptInvoice.pages
    WebCacheExport = myPDFExport.ExportWebCache(rptInvoice.pages)
End Function
```

Text Export Properties

Property	Data Type	Description
ByteOrderMark	Boolean	For unicode text, the export will place a byte order mark at the beginning of the file if true.
FileName	String	Specifies the name of the file to which the report will export.
PageDelimiter	String	Specifies the text inserted between each page.
SuppressEmptyLines	Boolean	Determines whether empty lines will be inserted for layout purposes.
TextDelimiter	String	Specifies the text delimiter to separate field items.
Unicode	Boolean	Determines whether the text will be saved as a UNICODE text file.

ByteOrderMark

Description

For unicode text, the export will place a byte order mark at the beginning of the file if true.

There is one spe0cial unicode code point, which identifies a unicode text file. This code point, is readable in iso-8859-1 is not part of the readable document. Text Export always outputs unicode in little-endian format. Always prefix a Unicode plain text file with a byte-order mark. Because Unicode plain text is a sequence of 16-bit code values, it is sensitive to the byte ordering used when the text was written. A byte-order mark is not a control character that selects the byte order of the text; it simply informs an application receiving the file that the file is byte ordered.

Ideally, all Unicode text would follow only one set of byte-ordering rules. This is not possible, however, because microprocessors differ in the placement of the least significant byte: IntelR and MIPSR processors position the least significant byte first, whereas Motorola processors (and all byte-reversed Unicode files) position it last. With only a single set of byte-ordering rules, users of one type of microprocessor would be forced to swap the byte order every time a plain text file is read from or written to, even if the file is never transferred to another system based on a different microprocessor. The preferred place to specify byte order is in a file header, but text files do not have headers. Therefore, Unicode has defined a character (0xFEFF) and a noncharacter (0xFFFE) as byte-order marks. They are mirror byte-images of each other. Since the sequence 0xFEFF is exceedingly rare at the outset of regular non-Unicode text files, it can serve as an implicit marker or signature to identify the file as a Unicode file.

Applications that read both Unicode and non-Unicode text files should use the presence of this sequence as an indicator that the file is most likely a Unicode file. (Compare this technique to using the MS-DOS EOF marker to terminate text files.) When an application finds 0xFEFF at the beginning of a text file, it typically processes the file as though it were a Unicode file, although it may also perform further heuristic checks to verify that this is true. Such a check could be as simple as testing whether the variation in the low-order bytes is much higher than the variation in the high-order bytes.

For example, if ASCII text is converted to Unicode text, every second byte is zero. Also, checking both for the linefeed and carriage-return characters (0x000A and 0x000D) and for even or odd file size can provide a strong indicator of the nature of the file. When an application finds 0xFFFE at the beginning of a text file, it interprets it to mean the file is a byte-reversed Unicode file. The application can either swap the order of the bytes or alert the user that an error has occurred. The Unicode byte-order mark character is not found in any code page, so it disappears if data is converted to ANSI. Unlike other Unicode characters, it is not replaced by a default character when it is converted. If a byte-order mark is found in the middle of a file, it is not interpreted as a Unicode character and has no effect on text output. The Unicode value 0xFFFF is illegal in plain text files and cannot be passed between Win32 functions. The value 0xFFFF is reserved for an application's private use.

DataType

Boolean

Settings

None

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    MyTXTExport.ByteOrderMark = True
    myTXTExport.PageDelimiter = ":"
```

```
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Export rptInvoice.pages

End Sub
```

PageDelimiter

Description

Sets or returns a text string used as a page delimiter.

Syntax

textExportObject.PageDelimiter=[Value]

Data Type

String

Settings

None

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    myTXTExport.PageDelimiter = ":"
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Export rptInvoice.pages

End Sub
```

SuppressEmptyLines

Description

Determines whether or not the exported file contains empty lines.

Syntax

textExportObject.SupressEmptyLines=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	The export filter will not output any empty lines to the report.
False	The export filter will output empty lines to the report.

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    myTXTExport.PageDelimiter = ":"
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Export rptInvoice.pages

End Sub
```

TextDelimiter

Description

Sets or returns a text string used to separate text fields in the filter output. The default setting for the TextDelimiter property is a tab character.

Note: Multi-line text may appear on separate lines in the exported text file. When dealing with multi-line layouts, an empty-string will help improve the exported results.

Syntax

textExportObject. TextDelimiter=[Value]

Data Type

String

Settings

None

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    myTXTExport.PageDelimiter = ":"
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Export rptInvoice.pages

End Sub
```

Unicode

Description

Specifies whether the text output will be in Unicode format as opposed to ASCII.

Syntax

```
textExportObject. Unicode=[Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	The report will output in Unicode format.
False	The report will output in ASCII format

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    myTXTExport.PageDelimiter = ":"
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Export rptInvoice.pages

End Sub
```

Text Export Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportStream	Exports a report's pages collection to a byte array. **Sub** ExportStream(*ActiveReportsPagesCollection* **As Unknown**, *OutputParamter*)

TIFF Export Properties

Property	Data Type	Description
FileName	String	Specifies the name of the file to which the report will export.

TIFF Export Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
FaxExport	Exports each page of the pages collection to an RFC 1314 TIFF image file. **Sub** FaxExport(p*agesColl* **As Unknown**, *Threshold* **As Long**)

Export

Description

Exports a pages collection to a TIFF file. The file is determined by the Filename property. The Export method requires a pages collection, which is created after a report, is run. You can call the Export method directly from the report object or from the viewer's pages collection. The TIFF file outputs images at 96 dpi with 8 bits of color depth for the 3 color channels. Any graphics or text visible in the margin area will be removed from the image. To calculate the TIFF width in pixels use the following for each canvas:

```
Dim Tiffwidth, leftmargintwips, rightmargintwips
Tiffwidth = (Canvas.Width – leftmargintwips – rightmargintwips) * 96 / 1440
```

Syntax

```
Sub Export(pagesColl as Unknown)
```

Parameters

Parameters

Name	Type	Description
pagesColl	IDispatch	The report's pages collection.

Example

```
Private Sub TextExport()
Dim myTXTExport As ActiveReportsTextExport.ARExportText
Set myTXTExport = New ActiveReportsTextExport.ARExportText

    myTXTExport.PageDelimiter = ":"
    myTXTExport.TextDelimiter = ";"
    myTXTExport.Unicode = True
    myTXTExport.SuppressEmptyLines = False
    myTXTExport.FileName = App.Path & "\TXTReport.txt"

    myTXTExport.Fax rptInvoice.pages

End Sub
```

FaxExport

Description
Exports the pages collection to a TIFF file, with the option of exporting a black and white image with halftone dithering or a black and white image where the white value corresponds to the threshold parameter. As the canvas is converted to monochrome, the sums of the colors' values are compared to the threshold value. If the sum of the colors is less than the threshold value, a black pixel will result. A threshold value of 766 would result in a totally black image.

Standard height for a fax compatible tiff is 1782 pixels by 2376 pixels. The export first checks to see if a page is landscape, then the export does a zoom calculation and attempts to keep the aspect ratio constant while scaling the canvas into the compatible tiff. The image width must be a multiple of 32.

Syntax

```
Sub FaxExport(pagesColl as Unknown, Threshold As Long)
```

Parameters

Name	Type	Description
pagesColl	IDispatch	A report's pages collection.
Threshold	Long	Sets the white value for black and white exports.

Example

```
Private Sub TextExport()
Dim myTIFFExport As ActiveReportsTIFFExport.TIFFExport
Set myTIFFExport = New ActiveReportsTIFFExport.TIFFExport

   myTIFFExport.FileName = app.path & "\TIFFExport.TIFF"
   myTIFFExport.FaxExport rptInvoice.pages 'Use halftone dithering

End Sub
```

HTML Export Properties

Property	Data Type	Description
AuxOutputPath	String	Destination path for images.
CharacterSet	String	Sets the destination character set.
CreateCSSFile	Boolean	Determines whether or not a .css file will be created when the report is exported.
CreateFramesetPage	Boolean	Determines if framesets will be created.
FilenamePrefix	String	Prefix for the output file's filename.
HTMLOutputPath	String	Sets the destination path for the exported HTML and MHT files.
HTMLVersion	DDHTMLVersion	Sets the HTML format version.
JPEGQuality	Long	Sets the JPEG compression quality as percent.
MHTOutput	Boolean	Determines if pages will be exported as MHT archives.
MultiPageOutput	Boolean	Determines whether or not the exported pages will be all on one HTML/MHT page.
TableOfContents	DDTableOfContents	Determines the type of style to use when exporting the table of contents.
Title	String	Sets the web page's title.

AuxOutputPath

Description

Destination path for images. Path is relative to the HTMLOutputPath.
Ex.
HTMLOutputPath = "C:\"
AuxOutputPath = "images"

Then images will be saved to "C:\images\" directory.

Syntax

```
htmlExportObject.AuxOutputPath=[Value]
```

Data Type

String

CharacterSet

Description

Sets the destination character set.

Syntax

```
htmlExportObject.CharacterSet=[Value]
```

Data Type

String

Supported Character Sets:

"UTF-8"
"UTF-7"
"UTF-7MIME" a different encoding of UTF-7 which outputs mail safe characters
"KOI8-r"
"KSC5601""
"shift-jis"
"iso-2022-jp"
"iso-2022-kr"
"euc-jp"
"big5"
"HZ-GB-2312"
"IBM850"
"iso-8859-1"
"iso-8859-2"
"iso-8859-5"
"iso-8859-6"

Three character sets have more than one alias. The first name is DDCharset name, the second is the Internet Explorer preferred name.

"cp1250" "x-cp1250" windows-1250 central european
"cp1251" "x-cp1251" windows-1251 cyrillic
"cp1252

CreateCSSFile

Description

Determines if a .css file will be created and exported to the HTMLOutputPath when the report is exported.

Syntax

```
htmlExportObject.CreateCSSFile=[Value]
```

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Create a .css file.
False	Does not create a .css file.

CreateFramesetPage

Description

Determines if a frameset and TOC frame will be generated. By default, no frameset and TOC frame will be generated. If CreateFramesetPage = True then the HTML file will be created as _.htm and the frameset will be .htm.

Syntax

htmlExportObject.CreateFramesetPage=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Create a HTML frameset page.
False	Does not create a HTML frameset page.

Example

```
'Creating a HTML Export Object

Dim Report as oYourReport
Dim oHTML As HTMLExport

Set Report = New oYourReport
Set oHTML= NewHTMLExport

'Set the FileNamePrefix equal to the String you want all HTML export file to
start with

'FileNamePrefix + HTMLOutPut are the only properties that are NOT option
oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'Setting HTMLVersion
oHTML.HTMLVersion = ddHVDHTML

'Setting CreateFramesPage property to True
oHTML.CreateFramesetPage = True

'Setting TablefContents property to one of the Enum values
oHTML.TableOfContents = ddTOCSimpleHTML

'Running Report
Report.run false

'Exporting Pages of Report to HTML output file

Report.Export oHTML
```

FileNamePrefix

Description

Sets the prefix for the output filename. The output filename will be combined with a page number if MultiPageOutput is True. Exporting with HTMLOutputPath = "C:\" and FileNamePrefix = "MyFile" will create C:\myname.htm.

Syntax

htmlExportObject.FileNamePrefix=[Value]

Data Type

String

Example

```
'Export the pages collection of a ARViewer Control

Dim oHTML As HTMLExport
Set oHTML = New HTMLExport

'Set the FileNamePrefix equal to the String you want all HTML files to start
with

oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'FileNamePrefix + HTMLOutputPath are the only properties that are NOT
optional
'Exporting Viewer Pages to HTML output file

oHTML.export Arviewer.pages
```

HTMLOutputPath

Description

Sets the destination path for the HTML and MHT files. Exporting with HTMLOutputPath = "C:\" and FileNamePrefix = "MyFile" will create C:\myname.htm.

Syntax

htmlExportObject.HTMLOutputPath=[Value]

Data Type

String

Example

```
'Export the pages collection of a ARViewer Control

Dim oHTML As HTMLExport
Set oHTML = New HTMLExport

'Set the FileNamePrefix equal to the String you want all HTML files to start
with

oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'FileNamePrefix + HTMLOutputPath are the only properties that are NOT
optional
'Exporting Viewer Pages to HTML output file

oHTML.export Arviewer.pages
```

HTMLVersion

Description

HTML format as DDHTMLVersion Enum

Syntax

```
htmlExportObject.HTMLVersion=[Value]
```

Data Type

DDHTMLVersion

Settings

Value	Mnemonic	Description
0	ddHVHTML32	Export HTML 3.2.
1	ddHVDHTML	Export HTML for CSS1 compliant browsers.

JPEGQuality

Description

Sets the JPEG compression quality as percent. Range 1-100.

Syntax

```
htmlExportObject.JPEGQuality=[Value]
```

Data Type

Long(Default = 100)

MHTOutput

Description

Determines if the pages will be exported as a MHT archive

Syntax

htmlExportObject.MHTOutput=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	Pages will be exported to a MHT archive.
False	Pages will not be exported to a MHT archive.

Example

```
'Creating a MHT Archive as multiple pages linked with a TOC page.
Dim oHTML As New HTMLExport


oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'Set MHTOutput to True so HTML Export will ouput a .MHT archive file
oHTML.MHTOutput = True

'Set JPEGQuality must be 1-100%
oHTML.JPEGQuality = 90

'Set MultiPageOutput to True so HTML Export Exports Each Page of the Report
to a seperate HTML file
oHTML.MultiPageOutput = True

'Set the Title Property of the HTML file
oHTML.Title = "Web Page"

'Exporting Report Pages 1-3 & 5 & 10 using ExportRange
oHTML.ExportRange Report.pages, "1-3,5,10"
```

MultiPageOutput

Description

Determines if the pages will be exported to one or multiple html pages.

Syntax

htmlExportObject.MultiPageOutput=[Value]

Data Type

Boolean (Default = False)

Settings

Value	Description
True	The export will create multiple pages.
False	The export will not create multiple pages.

Example

```
'Creating a MHT Archive as multiple pages linked with a TOC page.
Dim oHTML As New HTMLExport

oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'Set MHTOutput to True so HTML Export will ouput a .MHT archive file
oHTML.MHTOutput = True

'Set JPEGQuality must be 1-100%
oHTML.JPEGQuality = 90

'Set MultiPageOutput to True so HTML Export Exports Each Page of the Report
to a seperate HTML file
oHTML.MultiPageOutput = True

'Set the Title Property of the HTML file
oHTML.Title = "Web Page"

'Exporting Report Pages 1-3 & 5 & 10 using ExportRange
oHTML.ExportRange Report.pages, "1-3,5,10"
```

TableOfContents

Description

Determines if and how the table of contents will be exported. By default, no table of contents will be written.

Syntax

htmlExportObject.TableOfContents=[Value]

Data Type

DDTableOfContents

Settings

Value	Mnemonic	Description
0	ddTOCNone	The TOC will not be exported.
1	ddTOCSimpleHTML	Uses <DL> tag to display the TOC.
2	ddTOCDHTML	Outputs the TOC as a expandable DHTML tree.

Example

```
'Exporting to HTML with a Table of Contents navigation page.

Dim Report as oYourReport
Dim oHTML As HTMLExport

Set Report = New oYourReport
Set oHTML= NewHTMLExport

'Set the FileNamePrefix equal to the String you want all HTML export file to
start with

'FileNamePrefix + HTMLOutPut are the only properties that are NOT option
oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'Setting HTMLVersion
oHTML.HTMLVersion = ddHVDHTML

'Setting CreateFramesPage property to True
oHTML.CreateFramesetPage = True

'Setting TablefContents property to one of the Enum values
oHTML.TableOfContents = ddTOCSimpleHTML

'Running Report
Report.run false

'Exporting Pages of Report to HTML output file

Report.Export oHTML
```

Title

Description

Web page title.

Syntax

htmlExportObject.Title=[Value]

Data Type

String

HTML Export Methods

Method	Description
Export	Exports a report's pages collection to the document indicated by the Filename property. **Sub** Export(pagesColl as Unknown)
ExportRange	Expors a range of pages from the pages collection. **Sub** ExportRange(*pagesCol*l **As UnKnown**, [**pagesRange**], [*bStripHeaders* **As Boolean**])
ExportStream	Exports a report's pages collection to a byte array. **Sub** ExportStream(*ActiveReportsPagesCollection* **As Unknown**, *OutputParamter*)

ExportRange

Description

Exports a range of pages from the pages collection. The pagesRange parameter indicates which pages to export. “1,2,3-9,14”

Return Type

None

Syntax

```
Sub ExportRange(pagesColl as Unknown, [pagesRange], [bStripHeaders])
```

Parameters

Name	Type	Description
pagesColl	Variant	A report’s pages collection.
PagesRange	Variant	Indicates the page ranges to export.
bStripHeaders	Boolean	Strips the page header/footer from the exported report.

Example

```
Dim Report as oYourReport
Dim oHTML As ActiveReportsHTMLExport.HTMLexport

Set Report = New oYourReport
Set oHTML = New ActiveReportsHTMLExport.HTMLexport

'Set the FileNamePrefix property equal to the string you want all HTML files
to start with

oHTML.FileNamePrefix = "HTMLTest"
oHTML.HTMLOutPut Path = "C:\"

'Set the Title Property of the HTML file
oHTML.Title = "Web Page"

'Exporting Report Pages 1-3 & 5 & 10 using ExportRange
oHTML.ExportRange Report.Pages, "1-3,5,10"
```

HTML Export Events

Event	Description
ExportPageEnd	This event fires before the end of a page's processing. **Sub** ExportPageEnd(*IcurrentPage* **As Long**, *AddHTML* **As BSTR**)
ExportPageStart	This event fires before the start of a page's processing. **Sub** ExportPageStart(*IcurrentPage* **As Long**, *AddHTML* **As BSTR**)

ExportPageEnd

Description

This event fires before the end of a page's processing

Syntax

```
Sub ExportPageEnd (IcurrentPage as long, AddHTML As BSTR)
```

Parameters

Name	Description
ICurrentPage	The page currently being exported.
AddHTML	HTML String to add to the page being exported.

Example

```
Dim WithEvents myHTMLExport As ActiveReportsHTMLExport.HTMLexport

Private Sub myHTMLExport_ExportPageEnd(ByVal ICurrentPage As Long, AddHTML As
String)
    AddHTML = "<BR>Exported on " & Format(Date, "MM/DD/YYYY") & "<BR>"
End Sub
```

ExportPageStart

Description

This event fires before the start of a page's processing.

Syntax

```
Sub ExportPageStart (IcurrentPage as long, AddHTML As BSTR)
```

Parameters

Name	Description
ICurrentPage	The page currently being exported.
AddHTML	HTML String to add to the page being exported.

Example

```
Dim WithEvents myHTMLExport As ActiveReportsHTMLExport.HTMLexport

Private Sub myHTMLExport_ExportPageStart(ByVal ICurrentPage As Long, AddHTML
As String)
    AddHTML = "<BR>Exported on " & Format(Date, "MM/DD/YYYY") & "<BR>"
End Sub
```