
Show All

New Objects

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Publisher, including product news, technical articles, downloads, and samples.

The following table lists objects added to the Publisher object model.

Object	Description
AdvancedPrintOptions	Represents the advanced print settings for a publication.
BorderArt	Represents an available BorderArt.
BorderArtFormat	Represents the formatting of the BorderArt applied to the specified shape.
BorderArts	A collection of all BorderArt available for use in the specified publication.
CatalogMergeShapes	Represents the shapes contained in the catalog merge area of the specified publication.
ColorsInUse	A collection of **ColorFormat** objects that represent the colors present in the specified publication.
Documents	A collection that represents all open publications.
FindReplace	Represents the criteria for a find operation.
HeaderFooter	Represents the header or footer of a master page.
InlineShapes	A collection of **Shape** objects, which represent objects in the drawing layer, where **Shape.IsInline** is **True**.
Label	Represents a single unique label design available on the system.
Labels	A collection of **Label** objects, which represent the unique label designs available on the system.

PageBackground	Represents the background of a page.
PrintablePlate	Represents a single plate to be printed for the publication.
PrintablePlates	A collection of the **PrintablePlate** objects in a publication.
PrintableRect	Represents the sheet area within which the specified printer will print.
Section	Represents a Section of a publication or document.
Sections	A collection of all the **Section** objects in the document.
WebNavigationBarHyperlinks	A collection of all the **Hyperlink** objects of the specified **WebNavigationBarSet** object.
WebNavigationBarSet	Represents a Web navigation bar set for the current document.
WebNavigationBarSets	A collection of all the **WebNavigationBarSet** objects in the current document.
WebOptions	Represents the properties of a Web publication, including options for saving and encoding the publication, and enabling Web-safe fonts and font schemes.
WebPageOptions	Represents the properties of a single Web page within a Web publication, including options for adding the title and description of the page, background sounds, in addition to other options.

New Properties (Alphabetical List)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Publisher, including product news, technical articles, downloads, and samples.

The following table lists properties added to the Publisher object model (sorted alphabetically).

New Property	Object(s)
AddHebDoubleQuote	Options
AdvancedPrintOptions	Document
AllowBleeds	AdvancedPrintOptions
AlwaysSaveInDefaultEncoding	WebOptions
AutoUpdate	WebNavigationBarSet
AvailableLabels	PageSetup
Background	Page
BackgroundSound	WebPageOptions
BackgroundSoundLoopCount	WebPageOptions
BackgroundSoundLoopForever	WebPageOptions
BorderArt	Shape
BorderArts	Document
ButtonStyle	WebNavigationBarSet
CatalogMergeItems	Shape
CharBasedFirstLineIndent	ParagraphFormat
ColorModel	PictureFormat
ColorsInPalette	PictureFormat
ColorsInUse	Document
ColumnGutterWidth	LayoutGuides
ContinueNumbersFromPreviousSection	Section
Design	WebNavigationBarSet
Documents	Application

EffectiveResolution	PictureFormat
EmailAsImg	WebOptions
EnableIncrementalUpload	WebOptions
Encoding	WebOptions
Exists	BorderArtFormat, PageBackground
FieldType	MailMergeDataField
Filename	PictureFormat
FileSize	PictureFormat
Find	Document, TextRange
FindText	FindReplace
Footer	Page
Forward	FindReplace
FoundTextRange	FindReplace
GraphicsResolution	AdvancedPrintOptions
GutterCenterlines	LayoutGuides
HasAlphaChannel	PictureFormat
HasTransparencyColor	PictureFormat
Header	Page
HorizontalAlignment	WebNavigationBarSet
HorizontalBaseLineOffset	LayoutGuides
HorizontalBaseLineSpacing	LayoutGuides
HorizontalButtonCount	WebNavigationBarSet
HorizontalRepeat	CatalogMergeShapes
HorizontalScale	PictureFormat
ImageFormat	PictureFormat
IncludePageOnNewWebNavigationBars	WebPageOptions
InkName	Plate, PrintablePlate
InksToPrint	AdvancedPrintOptions
InlineAlignment	Shape, ShapeRange
InlineShapes	TextRange
InlineTextRange	Shape, ShapeRange
InUse	Plate

IsDataSourceConnected	Document
IsEmpty	PictureFormat
IsGreyScale	PictureFormat
IsHeader	HeaderFooter
IsHorizontal	WebNavigationBarSet
IsInline	Shape, ShapeRange
IsLeading	Page
IsLinked	PictureFormat
IsPostscriptPrinter	AdvancedPrintOptions
IsTrailing	Page
IsTrueColor	PictureFormat
IsTwoPageMaster	Page
IsWizard	Document
IsWizardPage	Page
KeepLinesTogether	ParagraphFormat
KeepWithNext	ParagraphFormat
Keywords	WebPageOptions
Label	PageSetup
LinkedFileStatus	PictureFormat
Links	WebNavigationBarSet
ListBulletFontName	ParagraphFormat
ListBulletFontSize	ParagraphFormat
ListBulletText	ParagraphFormat
ListIndent	ParagraphFormat
ListNumberSeparator	ParagraphFormat
ListNumberStart	ParagraphFormat
ListType	ParagraphFormat
LockToBaseLine	ParagraphFormat
MatchAlefHamza	FindReplace
MatchCase	FindReplace
MatchDiacritics	FindReplace
MatchKashida	FindReplace
MatchWholeWord	FindReplace

MatchWidth	FindReplace
NegativeImage	AdvancedPrintOptions
OrganizeInFolder	WebOptions
OriginalColorsInPalette	PictureFormat
OriginalFileSize	PictureFormat
OriginalHasAlphaChannel	PictureFormat
OriginalHeight	PictureFormat
OriginalIsTrueColor	PictureFormat
OriginalResolution	PictureFormat
OriginalWidth	PictureFormat
PageNumberFormat	Section
PageNumberStart	Section
PrintablePlates	AdvancedPrintOptions
PrintableRect	AdvancedPrintOptions
PrintBlankPlates	AdvancedPrintOptions
PrintBleedMarks	AdvancedPrintOptions
PrintColorBars	AdvancedPrintOptions
PrintCropMarks	AdvancedPrintOptions
PrintDensityBars	AdvancedPrintOptions
PrintJobInformation	AdvancedPrintOptions
PrintMode	AdvancedPrintOptions
PrintPageBackgrounds	Document
PrintPlate	PrintablePlate
PrintRegistrationMarks	AdvancedPrintOptions
PublicationLayout	PageSetup
PublicationType	Document
RedoActionsAvailable	Document
RelyOnVML	WebOptions
RemovePersonalInformation	Document
ReplaceScope	FindReplace
ReplaceWithText	FindReplace
Resolution	AdvancedPrintOptions

RowGutterWidth	LayoutGuides
Sections	Document
ShowHeaderFooterOnFirstPage	Section
ShowOnlyWebFonts	WebOptions
ShowSelected	WebNavigationBarSet
StartInNextTextBox	ParagraphFormat
StartPageIndex	Section
StretchPictures	BorderArtFormat
UndoActionsAvailable	Document
UseCharBasedFirstLineIndent	ParagraphFormat
UseCustomHalftone	AdvancedPrintOptions
UseOnlyPublicationFonts	AdvancedPrintOptions
VerticalBaseLineOffset	LayoutGuides
VerticalBaseLineSpacing	LayoutGuides
VerticalRepeat	CatalogMergeShapes
VerticalScale	PictureFormat
ViewHorizontalBaseLineGuides	Document
ViewVerticalBaseLineGuides	Document
WebNavigationBarSetName	Shape
WebNavigationBarSets	Document
WebOptions	Application
WebPageOptions	Page
WidowControl	ParagraphFormat

New Properties (by Object)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Publisher, including product news, technical articles, downloads, and samples.

The following table lists properties added to the Publisher object model (sorted by object name).

Object	New Properties
AdvancedPrintOptions	**AllowBleeds**, **GraphicsResolution**, **InksToPrint**, **IsPostscriptPrinter**, **NegativeImage**, **PrintablePlates**, **PrintableRect**, **PrintBlankPlates**, **PrintBleedMarks**, **PrintColorBars**, **PrintCropMarks**, **PrintDensityBars**, **PrintJobInformation**, **PrintMode**, **PrintRegistrationMarks**, **Resolution**, **UseCustomHalftone**, **UseOnlyPublicationFonts**
Application	**Documents**, **WebOptions**
BorderArtFormat	**Exists**, **StretchPictures**
CatalogMergeShapes	**HorizontalRepeat**, **VerticalRepeat**
Document	**AdvancedPrintOptions**, **BorderArts**, **ColorsInUse**, **Find**, **IsDataSourceConnected**, **IsWizard**, **PrintPageBackgrounds**, **PublicationType**, **RedoActionsAvailable**, **RemovePersonalInformation**, **Sections**, **UndoActionsAvailable**, **ViewHorizontalBaseLineGuides**, **ViewVerticalBaseLineGuides**, **WebNavigationBarSets**
FindReplace	**FindText**, **Forward**, **FoundTextRange**, **MatchAlefHamza**, **MatchCase**, **MatchDiacritics**, **MatchKashida**, **MatchWholeWord**, **MatchWidth**, **ReplaceScope**, **ReplaceWithText**
HeaderFooter	**IsHeader**

LayoutGuides	**ColumnGutterWidth**, **GutterCenterlines**, **HorizontalBaseLineOffset**, **HorizontalBaseLineSpacing**, **RowGutterWidth**, **VerticalBaseLineOffset**, **VerticalBaseLineSpacing**
MailMergeDataField	**FieldType**
Options	**AddHebDoubleQuote**
Page	**Background**, **Footer**, **Header**, **IsLeading**, **IsTrailing**, **IsTwoPageMaster**, **IsWizardPage**, **WebPageOptions**
PageBackground	**Exists**
PageSetup	**AvailableLabels**, **Label**, **PublicationLayout**
ParagraphFormat	**CharBasedFirstLineIndent**, **KeepLinesTogether**, **KeepWithNext**, **ListBulletFontName**, **ListBulletFontSize**, **ListBulletText**, **ListIndent**, **ListNumberSeparator**, **ListNumberStart**, **ListType**, **LockToBaseLine**, **StartInNextTextBox**, **UseCharBasedFirstLineIndent**, **WidowControl**
PictureFormat	**ColorModel**, **ColorsInPalette**, **EffectiveResolution**, **Filename**, **FileSize**, **HasAlphaChannel**, **HasTransparencyColor**, **HorizontalScale**, **ImageFormat**, **IsEmpty**, **IsGreyScale**, **IsLinked**, **IsTrueColor**, **LinkedFileStatus**, **OriginalColorsInPalette**, **OriginalFileSize**, **OriginalHasAlphaChannel**, **OriginalHeight**, **OriginalIsTrueColor**, **OriginalResolution**, **OriginalWidth**, **VerticalScale**
Plate	**InkName**, **InUse**
PrintablePlate	**InkName**, **PrintPlate**
Section	**ContinueNumbersFromPreviousSection**, **PageNumberFormat**, **PageNumberStart**, **ShowHeaderFooterOnFirstPage**, **StartPageIndex**
Shape	**BorderArt**, **CatalogMergeItems**, **InlineAlignment**, **InlineTextRange**, **IsInline**, **WebNavigationBarSetName**
ShapeRange	**InlineAlignment**, **InlineTextRange**, **IsInline**
TextRange	**Find**, **InlineShapes**

WebNavigationBarSet	**AutoUpdate**, **ButtonStyle**, **Design**, **HorizontalAlignment**, **HorizontalButtonCount**, **IsHorizontal**, **Links**, **ShowSelected**
WebOptions	**AlwaysSaveInDefaultEncoding**, **EmailAsImg**, **EnableIncrementalUpload**, **Encoding**, **OrganizeInFolder**, **RelyOnVML**, **ShowOnlyWebFonts**
WebPageOptions	**BackgroundSound**, **BackgroundSoundLoopCount**, **BackgroundSoundLoopForever**, **IncludePageOnNewWebNavigationBars**, **IncludePageOnWebNavigationBar**, **Keywords**

New Methods (Alphabetical List)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Publisher, including product news, technical articles, downloads, and samples.

The following table lists methods added to the Publisher object model (sorted alphabetically).

New Method	Object
AddCatalogMergeArea	Shapes
AddEmptyPictureFrame	Shapes
AddSet	WebNavigationBarSets
AddToCatalogMergeArea	Shape, ShapeRange
AddToEveryPage	WebNavigationBarSet
AddWebNavigationBar	Shapes
BeginCustomUndoAction	Document
ChangeOrientation	WebNavigationBarSet
ConvertPublicationType	Document
ConvertToProcess	Plate
Create	PageBackground
DeleteSetAndInstances	WebNavigationBarSet
EndCustomUndoAction	Document
ExportEmailHTML	Page
FindPlateByInkName	Plates, PrintablePlates
MoveIntoTextFlow	Shape, ShapeRange
MoveOutOfTextFlow	Shape, ShapeRange
Redo	Document
RemoveCatalogMergeArea	Shape
RemoveFromCatalogMergeArea	Shape, ShapeRange
Replace	PictureFormat
RevertToDefaultWeight	BorderArtFormat

RevertToOriginalColor	BorderArtFormat
Set	BorderArtFormat
SetBackgroundSoundRepeat	WebPageOptions
SetListType	ParagraphFormat
Undo	Document
WebPagePreview	Document

New Methods (by Object)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Publisher, including product news, technical articles, downloads, and samples.

The following table lists methods added to the Publisher object model (sorted by object name).

New Method	Object
BorderArtFormat	**RevertToDefaultWeight**, **RevertToOriginalColor**, **Set**
Document	**BeginCustomUndoAction**, **ConvertPublicationType**, **EndCustomUndoAction**, **Redo**, **Undo**, **WebPagePreview**
PageBackground	**Create**
Page	**ExportEmailHTML**
ParagraphFormat	**SetListType**
PictureFormat	**Replace**
Plate	**ConvertToProcess**
Plates	**FindPlateByInkName**
PrintablePlates	**FindPlateByInkName**
Shape	**AddToCatalogMergeArea**, **MoveIntoTextFlow**, **MoveOutOfTextFlow**, **RemoveCatalogMergeArea**, **RemoveFromCatalogMergeArea**
ShapeRange	**AddToCatalogMergeArea**, **MoveIntoTextFlow**, **MoveOutOfTextFlow**, **RemoveFromCatalogMergeArea**
Shapes	**AddCatalogMergeArea**, **AddEmptyPictureFrame**, **AddWebNavigationBar**
WebNavigationBarSet	**AddToEveryPage**, **ChangeOrientation**, **DeleteSetAndInstances**
WebNavigationBarSets	**AddSet**

Adjustments Object

Multiple objects └Adjustments

Contains a collection of adjustment values for the specified AutoShape or WordArt object. Each adjustment value represents one way an adjustment handle can be adjusted. Because some adjustment handles can be adjusted in two ways — for instance, some handles can be adjusted both horizontally and vertically — a shape can have more adjustment values than it has adjustment handles. A shape can have up to eight adjustments.

Using the Adjustments Object

Use the **Adjustments** property to return an **Adjustments** object. Use **Adjustments**(*index*), where *index* is the adjustment value's index number, to return a single adjustment value.

Different shapes have different numbers of adjustment values, different kinds of adjustments change the geometry of a shape in different ways, and different kinds of adjustments have different ranges of valid values.

The following table summarizes the ranges of valid adjustment values for different types of adjustments. In most cases, if you specify a value that's beyond the range of valid values, the closest valid value will be assigned to the adjustment.

Type of adjustment	Valid values
Linear (horizontal or vertical)	Generally the value 0.0 represents the left or top edge of the shape and the value 1.0 represents the right or bottom edge of the shape. Valid values correspond to valid adjustments you can make to the shape manually. For example, if you can only pull an adjustment handle half way across the shape manually, the maximum value for the corresponding adjustment will be 0.5. For shapes such as callouts, where the values 0.0 and 1.0 represent the limits of the rectangle defined by the starting and ending points of the callout line, negative numbers and numbers greater than 1.0 are valid values.
Radial	An adjustment value of 1.0 corresponds to the width of the shape. The maximum value is 0.5, or halfway across the shape.
Angle	Values are expressed in degrees. If you specify a value outside the range – 180 to 180, it will be normalized to be within that range.

The following example adds a right-arrow callout to the active document and sets adjustment values for the callout. Note that although the shape has only three adjustment handles, it has four adjustments. Adjustments three and four both correspond to the handle between the head and neck of the arrow.

```
Sub AdjustRightArrowCallout()
    With ActiveDocument.Pages(1).Shapes.AddShape( _
            Type:=msoShapeRightArrowCallout, Left:=72, Top:=72, _
            Width:=250, Height:=190).Adjustments
        .Item(1) = 0.75  'Adjusts width of text box
        .Item(2) = -0.5  'Adjusts width of arrowhead
        .Item(3) = 0.8   'Adjusts length of arrowhead
        .Item(4) = -0.75 'Adjusts width of arrow neck
    End With
End Sub
```

BorderArts Collection

Document └BorderArts
└BorderArt

A collection of all BorderArt available for use in the specified publication. BorderArt is predefined picture borders that can be applied to text boxes, picture frames, or rectangles.

Using the BorderArts Object

Use the **Item** property of a **BorderArts** collection to return a specific **BorderArt** object. The *Index* argument of the **Item** property can be the number or name of the BorderArt object.

This example returns the BorderArt "Apples" from the active publication.

```
Dim bdaTemp As BorderArt

Set bdaTemp = ActiveDocument.BorderArts.Item (Index:="Apples")
```

Use the **Count** property to return the number of BorderArt types available in the specified document. The following example displays the number of BorderArt types in the active document.

```
Sub CountBorderArts()
        MsgBox ActiveDocument.BorderArts.Count
End Sub
```

Remarks

The **BorderArts** collection includes any custom BorderArt types created by the user for the specified publication.


Show All

CatalogMergeShapes Collection

Shape └CatalogMergeShapes
└Shape
└Multiple objects

Represents the shapes contained in the catalog merge area of the specified publication.

Using the CatalogMergeShapes Collection

Use the **CatalogMergeItems** property of the **Shape** or **ShapeRange** objects to return the contents of the catalog merge area. The following example tests whether the specified publication contains a catalog merge area. If it does, it returns a list of the shapes it contains.

```
Sub ListCatalogMergeAreaContents()

    Dim pgPage As Page
    Dim mmLoop As Shape
    Dim intCount As Integer

    For Each pgPage In ThisDocument.Pages
        For Each mmLoop In pgPage.Shapes

            If mmLoop.Type = pbCatalogMergeArea Then

                With mmLoop.CatalogMergeItems
                    For intCount = 1 To .Count
                        Debug.Print "Shape ID: " & _
                            mmLoop.CatalogMergeItems.Item(intCount).
                        Debug.Print "Shape Name: " & _
                            mmLoop.CatalogMergeItems.Item(intCount).
                    Next
                End With

            End If

        Next mmLoop
    Next pgPage

End Sub
```

Use the **AddToCatalogMergeArea** method of the **Shape** or **ShapeRange** objects to add shapes to a catalog merge area. The following example adds a rectangle to the catalog merge area in the specified publication. This example assumes a catalog merge area has been added to the first page of the publication.

```
ThisDocument.Pages(1).Shapes.AddShape(1, 80, 75, 450, 125).AddToCata
```

Use **CatalogMergeItems**(*index*), where *index* is index number, to return a single catalog merge area shape. The following example removes the first shape from the catalog merge area.

```
ThisDocument.Pages(1).Shapes(1).CatalogMergeItems(1).RemoveFromCatal
```

Use the **RemoveFromCatalogMergeArea** method of the **Shape** or **ShapeRange** objects to remove shapes from a catalog merge area. Removed shapes are not deleted, but are instead placed on the publication page containing the catalog merge area. The following example tests whether the specified publication contains a catalog merge area. If it does, all the shapes are removed from the catalog merge area and deleted, and the catalog merge area is then removed from the publication.

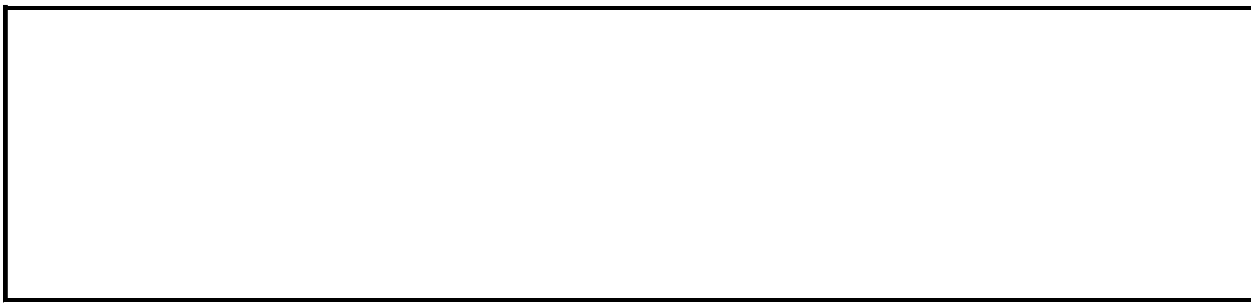
```
Sub DeleteCatalogMergeAreaAndAllShapesWithin()
    Dim pgPage As Page
    Dim mmLoop As Shape
    Dim intCount As Integer
    Dim strName As String

        For Each pgPage In ThisDocument.Pages
            For Each mmLoop In pgPage.Shapes

                If mmLoop.Type = pbCatalogMergeArea Then
                    With mmLoop.CatalogMergeItems
                        For intCount = .Count To 1 Step -1
                            strName = mmLoop.CatalogMergeItems.Item(
                            .Item(intCount).RemoveFromCatalogMergeAr
                            pgPage.Shapes(strName).Delete
                        Next
                    End With
                mmLoop.RemoveCatalogMergeArea
                End If

            Next mmLoop
        Next pgPage

 End Sub
```

Remarks

The catalog merge area is automatically resized to accommodate objects that are larger then the merge area, or that are positioned outside the catalog merge area when they are added.

Shapes inside the catalog merge area are automatically resized or repositioned if the catalog merge area is decreased in size or moved.

The catalog merge area can contain picture and text data fields you have inserted, as well as other design elements you choose.



CellRange Collection

Multiple objects └CellRange
└Cell
└Multiple objects

A collection of **Cell** objects in a table column or row. The **CellRange** collection represents all the cells in the specified column or row.

Using the CellRange Collection

Use the **Cells** property to return the **CellRange** collection. This example merges the cells in first column of the table.

```
Sub MergeCellsInFirstColumn()
    With ActiveDocument.Pages(1).Shapes(1).Table
        .Cells(StartRow:=1, StartColumn:=1, _
        EndRow:=.Rows.Count, EndColumn:=1).Select
    End With
    Selection.TableCellRange.Merge
End Sub
```

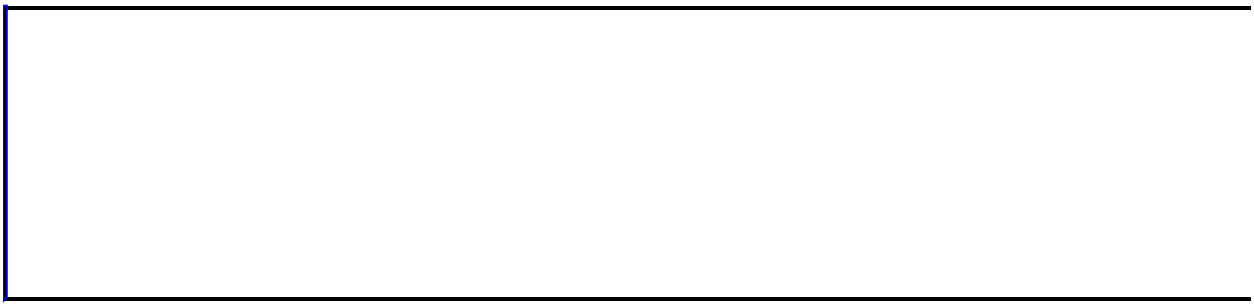
Use the **Count** property to return the number of cells in a row, column, table or selection. This example displays a message with the number of cells the specified table.

```
Sub NumberOfTableCells()
    MsgBox ActiveDocument.Pages(1).Shapes(1).Table _
        .Cells.Count
End Sub
```

Remarks

Although the collection object is named **CellRange** and is shown in the Object Browser, this keyword is not used in programming the Microsoft Publisher object model. The keyword **Cells** is used instead.

You cannot programmatically add to or delete individual cells from a Publisher table. Use the **AddTable** method with the **Shapes** collection to add a new table to a publication. Use the **Add** method of the **Columns** or **Rows** collections to add a column or row to a table. Use the **Delete** method of the **Columns** or **Rows** collections to delete a column or row from a table.



ColorSchemes Collection

Application └ColorSchemes
└ColorScheme
└ColorFormat

A collection of all the **ColorScheme** objects in Microsoft Publisher. Each **ColorScheme** object represents a color scheme, which is a set of colors that are used in a publication.

Using the ColorSchemes collection

Use the **Count** property to return the number of color schemes available to Microsoft Publisher. The following example displays the number of color schemes.

```
Sub CountColorSchemes()
    MsgBox Application.ColorSchemes.Count
End Sub
```

Use the **Item** property to return a specific color scheme from the **ColorSchemes** collection. The ***Index*** argument of the **Item** property can be the number or name of the color scheme, or a **PbColorScheme** constant. The follow example sets the color scheme of the active publication to Wildflower.

```
Sub SetColorScheme()
    ActiveDocument.ColorScheme _
        = ColorSchemes.Item(pbColorSchemeWildflower)
End Sub
```

Use the **Name** property to return a color scheme name. The following example lists in a text box all the color schemes available to Publisher.

```
Sub ListColorShemes()

    Dim clrScheme As ColorScheme
    Dim strSchemes As String

    For Each clrScheme In Application.ColorSchemes
        strSchemes = strSchemes & clrScheme.Name & vbLf
    Next
    ActiveDocument.Pages(1).Shapes.AddTextbox( _
        Orientation:=pbTextOrientationHorizontal, _
        Left:=72, Top:=72, Width:=400, Height:=500).TextFrame _
        .TextRange.Text = strSchemes

End Sub
```


Show All

ColorsInUse Collection

Document └ColorsInUse
└ColorFormat
└ColorCMYK

A collection of **ColorFormat** objects that represent the colors present in the specified publication.

Using the ColorsInUse Object

Use the **ColorsInUse** property of the **Document** object to return the **ColorsInUse** collection.

The following example lists properties of each color in the active publication that is based on the specified ink. This example assumes the publication's color mode has been defined as spot color or process and spot color.

```
Sub ListColorsBasedOnInk()
Dim cfLoop As ColorFormat

For Each cfLoop In ActiveDocument.ColorsInUse

    With cfLoop
        If .Ink = "2" Then
            Debug.Print "BaseRGB: " & .BaseRGB
            Debug.Print "RGB: " & .RGB
            Debug.Print "TintShade: " & .TintAndShade
            Debug.Print "Type: " & .Type
        End If
    End With

Next cfLoop

End Sub
```

Use **ColorsInUse**(*index*), where *index* is the color index number, to return a single **ColorFormat** object. The following example returns properties for the second color in the publication.

```
Sub ColorProperties()

    With ActiveDocument.ColorsInUse(2)
        Debug.Print "Color RBG: " & .RGB
        Debug.Print "Ink RBG: " & .BaseRGB
        Debug.Print "Tint: " & .TintAndShade

    End With

End Sub
```

Remarks

The **ColorsInUse** collection supports all the publication color models: RGB, process colors, and spot color.

For process color and spot color publications, colors are based on inks. For a given ink, a publication may contain several colors that are different tints or shades of that ink. Use the **Plates** collection to access the plates that represent the inks defined for a publication.

Columns Collection

Table
└Columns
└column
└CellRange

A collection of **Column** objects that represent the columns in a table.

Using the Columns collection

Use the **Columns** property of the **Table** object to return the **Columns** collection. The following example displays the number of **Column** objects in the **Columns** collection for the first table in the active document.

```
Sub CountColumns()
    MsgBox "The number of columns in the table is " & _
        ActiveDocument.Pages(2).Shapes(1).Table.Columns.Count
End Sub
```

This example enters a bold number into each cell in the specified table. This example assumes the specified shape is a table and not another type of shape.

```
Sub CountCellsByColumn()
    Dim shpTable As Shape
    Dim colTable As Column
    Dim celTable As Cell
    Dim intCount As Integer

    intCount = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each colTable In shpTable.Table.Columns
        For Each celTable In colTable.Cells
            With celTable.Text
                .Text = intCount
                .ParagraphFormat.Alignment = _
                    pbParagraphAlignmentCenter
                .Font.Bold = msoTrue
                intCount = intCount + 1
            End With
        Next celTable
    Next colTable

End Sub
```

Use **Columns**(*index*), where *index* is the index number, to return a single **Column** object. The index number represents the position of the column in the **Columns** collection (counting from left to right). The following example selects the third column in the specified table.

```
Sub SelectColumns()
    ActiveDocument.Pages(2).Shapes(1).Table.Columns(3).Cells.Select
End Sub
```

Use the **Add** method to add a column to a table. This example adds a column to the specified table on the second page of the active publication, and then adjusts the width, merges the cells, and sets the fill color. This example assumes the first shape is a table and not another type of shape.

```
Sub NewColumn()
    Dim colNew As Column

    Set colNew = ActiveDocument.Pages(2).Shapes(1).Table.Columns _
        .Add(BeforeColumn:=3)
    With colNew
        .Width = 2
        .Cells.Merge
        .Cells(1).Fill.ForeColor.RGB = RGB(Red:=202, Green:=202, Blu
    End With
End Sub
```

Documents Collection

Application └Documents
└Document
└Multiple objects

Represents all open publications. The **Documents** collection contains all **Document** objects that are open in Publisher.

Using the Documents Collection

Use the **Documents** property to return the **Documents** collection. The following example lists all of the open publications.

```
Dim objDocument As Document
Dim strMsg As String
For Each objDocument In Documents
    strMsg = strMsg & objDocument.Name & vbCrLf
Next objDocument
MsgBox Prompt:=strMsg, Title:="Current Documents Open", Buttons:=vbC
```

Use the **Add** method to add a new document to the collection. A new and visible instance of Publisher is created when the **Add** method is called. The following example adds a new document to the **Documents** collection.

```
Dim objDocument As Document
Set objDocument = Documents.Add
With objDocument
    .LayoutGuides.Columns = 4
    .LayoutGuides.Rows = 9
    .ActiveView.Zoom = pbZoomWholePage
End With
```

Use the **Item**(*index*) property, where *index* is the index number or document name as a **String**, to return a specific document object. The following example displays the name of the first open publication.

```
If Documents.Count >= 1 Then
    MsgBox Documents.Item(1).Name
End If
```

The following example checks the name of each document in the **Documents** collection. If the name of a document is "sales.doc", an object variable objSalesDoc is set to that document in the **Documents** collection.

```
Dim objDocument As Document
Dim objSalesDoc As Document
For Each objDocument In Documents
    If objDocument.Name = "sales.pub" Then
        Set objSalesDoc = objDocument
    End If
Next objDocument
```

Fields Collection

TextRange └Fields
└Field
└Multiple objects

A collection of **Field** objects that represent all the fields in a text range.

Using the Fields Collection

Use the **Fields** property to return the **Fields** collection. Use **Fields**(*index*), where *index* is the index number, to return a single **Field** object. The index number represents the position of the field in the selection, range, or publication. The following example displays the field code and the result of the first field in each text box in the active publication.

```
Sub ShowFieldCodes()
    Dim pagPage As Page
    Dim shpShape As Shape

    For Each pagPage In ActiveDocument.Pages
        For Each shpShape In pagPage.Shapes
            If shpShape.Type = pbTextFrame Then
                With shpShape.TextFrame.TextRange
                    If .Fields.Count > 0 Then
                        MsgBox "Code =  " & .Fields(1).Code & vbLf _
                            & "Result =  " & .Fields(1).Result & vbL
                    End If
                End With
            End If
        Next
    Next
End Sub
```

The **Count** property for this collection in a publication returns the number of items in a specified shape or selection.

GroupShapes Collection

Multiple objects └GroupShapes
└Shape
└Multiple objects

Represents the individual shapes within a grouped shape. Each shape is represented by a **Shape** object. Using the **Item** method with this object, you can work with single shapes within a group without having to ungroup them.

Using The Groupshapes Collection

Use the **GroupItems** property to return a **GroupShapes** collection. Use **GroupItems**(*index*), where *index* is the number of the individual shape within the grouped shape, to return a single shape from the **GroupShapes** collection. The following example adds three triangles to the active document, groups them, sets a color for the entire group, and then changes the color for the third triangle only.

```
Sub WorkWithGroupShapes()
    With ActiveDocument.Pages.Add(Count:=1, After:=1).Shapes
        .AddShape(msoShapeIsoscelesTriangle, _
            50, 50, 100, 100).Name = "shpOne"
        .AddShape(msoShapeIsoscelesTriangle, _
            200, 50, 100, 100).Name = "shpTwo"
        .AddShape(msoShapeIsoscelesTriangle, _
            350, 50, 100, 100).Name = "shpThree"
        With .Range(Array("shpOne", "shpTwo", "shpThree")).Group
            .Fill.PresetTextured PresetTexture:=msoTextureBlueTissue
            .GroupItems(3).Fill.PresetTextured _
                PresetTexture:=msoTextureGreenMarble
        End With
    End With
End Sub
```

Hyperlinks Collection

TextRange └Hyperlinks
└Hyperlink
└Multiple objects

Represents the collection of **Hyperlink** objects in a text range.

Using the Hyperlinks Collection

Use the **Hyperlinks** property to return the **Hyperlinks** collection. The following example deletes all text hyperlinks in the active publication that contain the word "Tailspin" in the address.

```
Sub DeleteMSHyperlinks()
    Dim pgsPage As Page
    Dim shpShape As Shape
    Dim hprLink As Hyperlink
    For Each pgsPage In ActiveDocument.Pages
        For Each shpShape In pgsPage.Shapes
            If shpShape.HasTextFrame = msoTrue Then
                If shpShape.TextFrame.HasText = msoTrue Then
                    For Each hprLink In shpShape.TextFrame.TextRange
                        If InStr(hprLink.Address, "tailspin") <> 0 T
                            hprLink.Delete
                            Exit For
                        End If
                    Next
                Else
                    shpShape.Hyperlink.Delete
                End If
            End If
        Next
    Next
End Sub
```

Use the **Add** method to create a hyperlink and add it to the **Hyperlinks** collection. The following example creates a new hyperlink to the specified Web site.

```
Sub AddHyperlink()
    Selection.TextRange.Hyperlinks.Add Text:=Selection.TextRange, _
    Address:="http://www.tailspintoys.com/"
End Sub
```

Use **Hyperlinks**(*index*), where *index* is the index number, to return a single **Hyperlink** object in a publication, range, or selection. This example displays the address for the first hyperlink if the specified selection contains hyperlinks.

```
Sub DisplayHyperlinkAddress()
```

```
    With Selection.TextRange.Hyperlinks
        If .Count > 0 Then _
            MsgBox .Item(1).Address
    End With
End Sub
```

The **Count** property for this collection returns the number of hyperlinks in the specified shape or selection only.

InlineShapes Collection

TextRange └InlineShapes
└ShapeRange

Contains a collection of **Shape** objects, which represent objects in the drawing layer, where **Shape.IsInline** is **True**. The collection of shapes is limited to shapes within a given text range.

Using the InlineShapes Collection

Use the **InlineShapes** property on the **TextRange** object to return an **InlineShapes** collection. The following example finds the first shape, a text box, on page one of the publication, and appends text to the end of the text range in the text box if there is more than one inline shape within the text range.

```
Dim theShape As Shape

Set theShape = ActiveDocument.Pages(1).Shapes(1)

With theShape.TextFrame.TextRange
    If .InlineShapes.Count > 1 Then
        .InsertAfter (" There is more than one inline shape in this
    End If
End With
```

Use the **InlineShapes**(*index*) property to return a single inline shape. The following example finds the third inline shape within a text box and flips it vertically.

```
Dim theShape As Shape

Set theShape = ActiveDocument.Pages(1).Shapes(1)

With theShape.TextFrame.Story.TextRange
    With .InlineShapes(3)
        .Flip (msoFlipVertical)
    End With
End With
```

Use the **Range** property to return a **ShapeRange** object that contains all members of the **InlineShapes** collection. An array of indexes or strings or a single index or string can be passed as a parameter of the **Range** property to select particular shapes or a shape within the range. The following example sets a **ShapeRange** variable equal to the collection of inline shapes that exist within a text box. Each inline shape within the range is then modified in some way. This example assumes that the first shape on the page is a text box that contains three inline shapes.

```
Dim theRange As ShapeRange

Set theRange = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.Story.TextRange.InlineShapes.Range

With theRange
    .Item(1).Flip msoFlipVertical
    .Item(2).MoveOutOfTextFlow
    .Item(3).Delete
End With
```

Remarks

The **InlineShapes** collection is available only on the **TextRange** object. Using **TextFrame.Story.TextRange.InlineShapes** will return all inline shapes in a text frame, including those that are in overflow. Using **TextFrame.TextRange.InlineShapes** will return only visible inline shapes in a text frame, and not those that are in overflow.

The **InlineShapes** collection can also be accessed from **Document.Stories(*i*).TextRange**, where *i* is the index to the active page of the publication.

The **InlineShapes** collection is not available in the **Page.Shapes** collection, including its contained **ShapeRange**.

Labels Collection

PageSetup └Labels
└Label

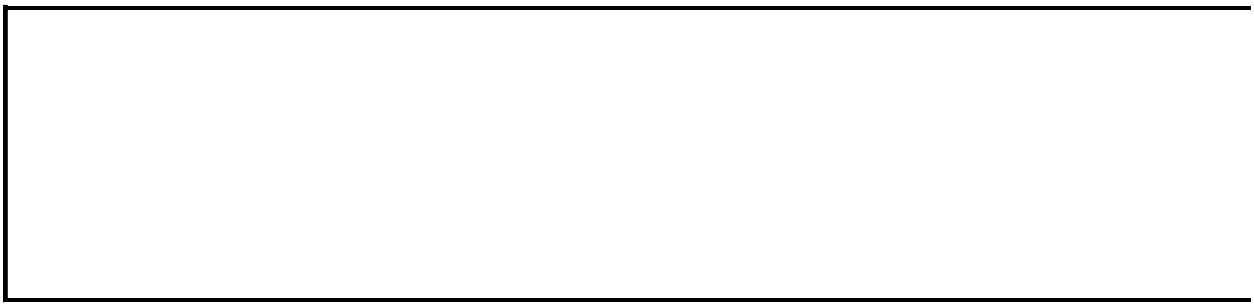
Contains a collection of **Label** objects, which represent the unique label designs available on the system.

Using the Labels collection

Each label design available on the system resides in the **AvailableLabels** collection. Use the **AvailableLabels** property on the **PageSetup** object to return the collection of **Label** objects that are available on the system.

The following example uses the **AvailableLabels** property to populate the **Labels** collection with the lables that are available in the active document. A test is then run on each label in the collection, and the name of the label is displayed if the label's height is greater than 4 inches.

```
Dim theLabel As Label
Dim theLabels As Labels

Set theLabels = ActiveDocument.PageSetup.AvailableLabels

For Each theLabel In theLabels
    If theLabel.Height > InchesToPoints(4) Then
        MsgBox theLabel.Name
    End If
Next theLabel
```




Show All

MailMergeDataFields Collection

MailMergeDataSource └MailMergeDataFields
└MailMergeDataField

A collection of **MailMergeDataField** objects that represent the data fields in a mail merge or catalog merge data source.

Using the MailMergeDataFields Collection

Use the **DataFields** property to return the **MailMergeDataFields** collection.

The following example displays the field names in the data source attached to the active publication.

```
Sub ShowFieldNames()
    Dim intCount As Integer
    With ActiveDocument.MailMerge.DataSource.DataFields
        For intCount = 1 To .Count
            MsgBox .Item(intCount).Name
        Next
    End With
End Sub
```

Use **DataFields**(*index*), where *index* is the data field name or the index number, to return a single **MailMergeDataField** object. The index number represents the position of the data field in the mail merge data source. This example retrieves the name of the first field and value of the first record of the FirstName field in the data source attached to the active publication.

```
Sub GetDataFromSource()
    With ActiveDocument.MailMerge.DataSource.DataFields
        MsgBox "First field name: " & .Item(1).Name & vbLf & _
            "Value of the first record of the FirstName field: " & _
            .Item("FirstName").Value
    End With
End Sub
```

Remarks

You cannot add fields to the **MailMergeDataFields** collection. When a data field is added to a data source, the field is automatically included in the **MailMergeDataFields** collection.


Show All

MailMergeFilters Collection

MailMergeDataSource └MailMergeFilters

Represents all the filters to apply to the data source attached to the mail merge or catalog merge publication. The **MailMergeFilters** object is comprised of **MailMergeFilterCriterion** objects.

Using the MailMergeFilters object

Use the **Add** method of the **MailMergeFilters** object to add a new filter criterion to the query. This example adds a new line to the query string and then applies the combined filter to the data source. This example assumes that a data source is attached to the active publication.

```
Sub FilterDataSource()
    With ActiveDocument.MailMerge.DataSource
        .Filters.Add Column:="Region", _
            Comparison:=msoFilterComparisonIsBlank, _
            Conjunction:=msoFilterConjunctionAnd
        .ApplyFilter
    End With
End Sub
```

Use the **Item** method to access an individual filter criterion. This example loops through all the filter criterion and if it finds one with a value of "Region", changes it to remove from the mail merge all records that are not equal to "WA". This example assumes that a data source is attached to the active publication.

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next
    End With
End Sub
```

MailMergeMappedDataFields Collection

MailMergeDataSource └MailMergeMappedDataFields
└MailMergeMappedDataField

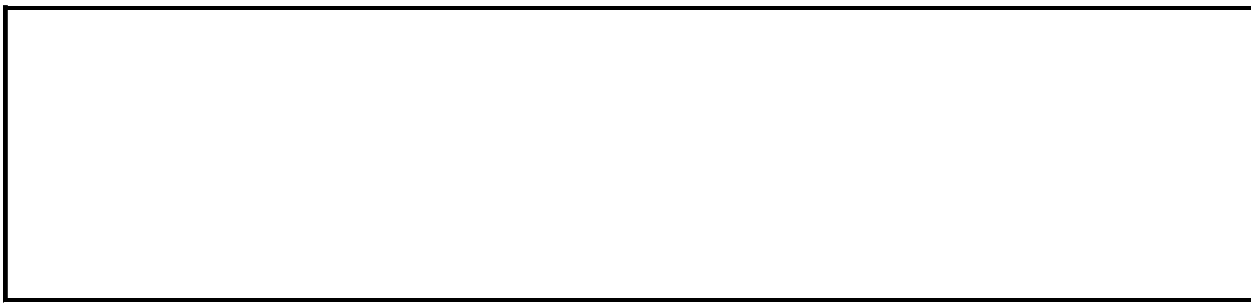
A collection of **MailMergeMappedDataField** objects that represents the mapped data fields available in Publisher.

Using the MailMergeMappedDataFields collection

Use the **MappedDataFields** property of the **MailMergeDataSource** object to return the **MailMergeMappedDataFields** collection. This example creates a table on a new page of the current publication and lists the mapped data fields available in Publisher and the fields in the data source to which they are mapped. This example assumes that the current publication is a mail merge publication and that the data source fields have corresponding mapped data fields.

```
Sub MappedFields()
    Dim intCount As Integer
    Dim intRows As Integer
    Dim docPub As Document
    Dim pagNew As Page
    Dim shpTable As Shape
    Dim tblTable As Table
    Dim rowTable As Row

    On Error Resume Next

    Set docPub = ThisDocument
    Set pagNew = ThisDocument.Pages.Add(Count:=1, After:=1)
    intRows = docPub.MailMerge.DataSource.MappedDataFields.Count + 1

    'Creates new table with a heading row
    Set shpTable = pagNew.Shapes.AddTable(NumRows:=intRows, _
        numColumns:=2, Left:=100, Top:=100, Width:=400, Height:=12)
    Set tblTable = shpTable.Table
    With tblTable.Rows(1)
        With .Cells(1).Text
            .Text = "Mapped Data Field"
            .Font.Bold = msoTrue
        End With
        With .Cells(2).Text
            .Text = "Data Source Field"
            .Font.Bold = msoTrue
        End With
    End With

    With docPub.MailMerge.DataSource
        For intCount = 2 To intRows - 1
            'Inserts mapped data field name and the
            'corresponding data source field name
            tblTable.Rows(intCount - 1).Cells(1).Text _
                .Text = .MappedDataFields(Index:=intCount).Name
```

```
            tblTable.Rows(intCount - 1).Cells(2).Text _
                .Text = .MappedDataFields(Index:=intCount).DataFiel
        Next
    End With
End Sub
```



MasterPages Collection

Document └MasterPages
└Page
└Multiple objects

Represents the page master for a publication after which all pages in the publication will be designed. The **MasterPages** object is a collection of **Page** objects.

Using the MasterPages collection

Use the **MasterPages** property to return a **MasterPages** object. The following example adds two ruler guides to the master page so that each page in the active publication is divided into quarters.

```
Sub ChangeMasterPage()
        Dim intWidth As Integer
    Dim intHeight As Integer

    With ActiveDocument
        intWidth = .PageSetup.PageWidth
        intWidth = intWidth / 2
        intHeight = .PageSetup.PageHeight
        intHeight = intHeight / 2
        With .MasterPages(1).RulerGuides
            .Add Position:=intWidth, _
                Type:=pbRulerGuideTypeVertical
            .Add Position:=intHeight, _
                Type:=pbRulerGuideTypeHorizontal
        End With
    End With
End Sub
```

Use the **Shapes** property to work with AutoShapes and text boxes on the master page. This example adds a small red heart shape to the upper left corner of the master page that will appear on each page in the active publication.

```
Sub AddShapeToMasterPage()
    ActiveDocument.MasterPages(1).Shapes.AddShape(Type:=msoShapeHear
        Left:=36, Top:=36, Width:=36, Height:=36).Fill _
        .ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)
End Sub
```

ObjectVerbs Collection

OLEFormat └ObjectVerbs

Represents the collection of OLE verbs for the specified OLE object. OLE verbs are the operations supported by an OLE object. Commonly used OLE verbs are play and edit.

Using the ObjectVerbs object

Use the **ObjectVerbs** property to return an **ObjectVerbs** object. The following example displays all the available verbs for the OLE object contained in the first shape on first page in the active publication. For this example to work, the specified shape must contain an OLE object.

```
Sub GetVerbs()
    Dim intCount As Integer

    With ActiveDocument.Pages(1).Shapes(1).OLEFormat
        For intCount = 1 To .ObjectVerbs.Count
            MsgBox .ObjectVerbs(intCount)
        Next
    End With
End Sub
```

Pages Collection

Document └Pages
└Page
└Multiple objects

Represents all the pages in a publication. The **Pages** collection contains all the **Page** objects in a publication.

Using the Pages collection

Use the **Add** method to add a new page to a publication. The following example adds a new page and a shape to the active publication.

```
Sub AddPageAndShape()
    With ActiveDocument.Pages.Add(Count:=1, After:=1)
        With .Shapes.AddShape(Type:=msoShape5pointStar, _
                Left:=72, Top:=72, Width:=50, Height:=50)
            .Fill.ForeColor.RGB = RGB(Red:=128, Green:=50, Blue:=255
            .Line.ForeColor.RGB = RGB(Red:=75, Green:=50, Blue:=255)
        End With
    End With

End Sub
```

Plates Collection

Document └Plates
└Plate
└ColorFormat

A collection of **Plate** objects in a publication.

Using the Plates collection

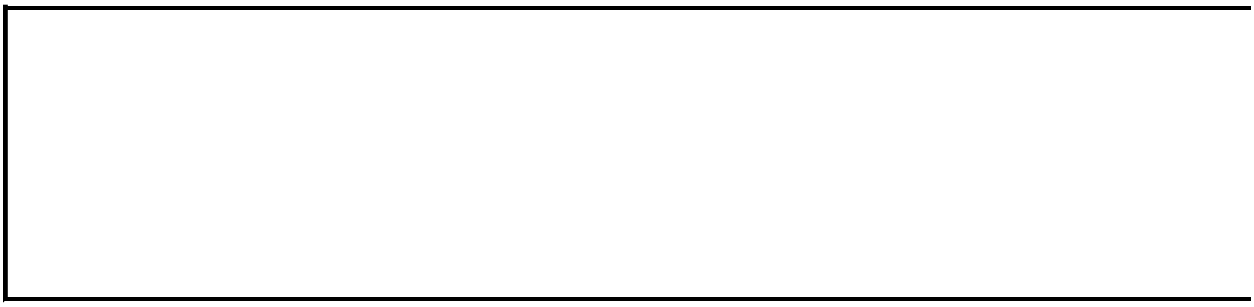
The **Plates** collection is made up of **Plate** objects for the various publication color modes. Each publication can only use one color mode. For example, you can't specify the spot-color mode in a procedure and then later specify the process-color mode. Use the **CreatePlateCollection** method of the **Document** object to specify which color mode to use in a publication's plate collection. Use the **Add** method of the **Plates** collection to add a new plate to the **Plates** collection. This example creates a new spot-color plate collection and adds a plate to it.

```
Sub AddNewPlates()
    Dim plts As Plates
    Set plts = ActiveDocument.CreatePlateCollection(Mode:=pbColorMod
    plts.Add
    With plts(1)
        .Color.RGB = RGB(Red:=255, Green:=0, Blue:=0)
        .Luminance = 4
    End With
End Sub
```

Use the **EnterColorMode** method of the **Document** object to the specify the color mode and the **Plates** collection to use with the color mode. Use the **ColorMode** property to determine which color mode is in use in a publication. This example creates a spot-color plate collection, adds two plates to it, and then enters those plates into the spot-color mode.

```
Sub CreateSpotColorMode()
    Dim plArray As Plates

    With ThisDocument
        'Creates a color plate collection,
        'which contains one black plate by default
        Set plArray = .CreatePlateCollection(Mode:=pbColorModeSpot)

        'Sets the plate color to red
        plArray(1).Color.RGB = RGB(255, 0, 0)

        'Adds another plate, black by default and
        'sets the plate color to green
        plArray.Add
        plArray(2).Color.RGB = RGB(0, 255, 0)
```

```
        'Enters spot-color mode with above
        'two plates in the plates array
        .EnterColorMode Mode:=pbColorModeSpot, Plates:=plArray
    End With
End Sub
```



PrintablePlates Collection

AdvancedPrintOptions └PrintablePlates
└PrintablePlate

A collection of the **PrintablePlate** objects in a publication.

Using the PrintablePlates collection

Use the **PrintablePlates** property of the **AdvancedPrintOptions** object to return the **PrintablePlates** collection. The following example returns a list of the printable plates currently in the collection for the active document. The example assumes that separations have been specified as the active publication's print mode.

```
Sub ListPrintablePlates()
    Dim pplTemp As PrintablePlates
    Dim pplLoop As PrintablePlate


    Set pplTemp = ActiveDocument.AdvancedPrintOptions.PrintablePlate
    Debug.Print "There are " & pplTemp.Count & " printable plates in

    For Each pplLoop In pplTemp
        With pplLoop
            Debug.Print "Printable Plate Name: " & .Name
            Debug.Print "Index: " & .Index
            Debug.Print "Ink Name: " & .InkName
            Debug.Print "Plate Angle: " & .Angle
            Debug.Print "Plate Frequency: " & .Frequency
            Debug.Print "Print Plate?: " & .PrintPlate
        End With
    Next pplLoop
End Sub
```

Use the **FindPlateByInkName** method to return a specific plate by referencing its ink name. The following example returns a spot color plate and sets several of its properties. The example assumes that separations have been specified as the active publication's print mode.

```
Sub SetPlatePropertiesByInkName()

Dim pplPlate As PrintablePlate
ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

    Set pplPlate = ActiveDocument.AdvancedPrintOptions.PrintablePlat

    With pplPlate
        .Angle = 75
        .Frequency = 133
```

```
        .PrintPlate = True
    End With

End Sub
```

Remarks

The **PrintablePlates** collection is generated when a publication's print mode is set to separations. Returns "Permission Denied" when any other print mode is specified.

The **PrintablePlates** collection represents the plates that will actually be printed for the publication, based on:

- The plates (if any) you have defined for the publication
- The advanced print options specified

You cannot programmatically create a printable plates collection, or add a printable plate to the collection.

Use the following properties of the **AdvancedPrintOptions** object to specify which plates are included in the **PrintablePlates** collection:

- Use the **PrintMode** property to set the publication to print as separations.
- Use the **InksToPrint** property to select which types of plates to print.
- Use the **PrintPlate** property to select individual plates to print.
- Use the **PrintBlankPlates** to determine whether to print plates for any pages where an ink is not used.

This collection corresponds to the plates listed on the **Separations** tab of the **Advanced Print Settings** dialog box.

Rows Collection

Table
└Rows
└Row
└CellRange

A collection of **Row** objects that represent the rows in a table.

Using the Rows collection

Use the **Rows** property of the **Table** object to return the **Rows** collection. The following example displays the number of **Row** objects in the **Rows** collection for the first table in the active document.

```
Sub CountRows()
    MsgBox ActiveDocument.Pages(2).Shapes(1).Table.Rows.Count
End Sub
```

This example sets the fill for all even-numbered rows and clears the fill for all odd-numbered rows in the specified table. This example assumes the specified shape is a table and not another type of shape.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            If celTable.Row Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
        Next celTable
    Next rowTable

End Sub
```

Use **Rows**(*index*), where *index* is the index number, to return a single **Row** object. The index number represents the position of the row in the **Rows** collection (counting from left to right). The following example selects the third row in the specified table.

```
Sub SelectRows()
    ActiveDocument.Pages(2).Shapes(1).Table.Rows(3).Cells.Select
End Sub
```

RulerGuides Collection

Page └RulerGuides
└RulerGuide

A collection of **RulerGuide** objects that represents a grid line used to align objects on a page.

Using the RulerGuides collection

Use the **Add** method of the **RulerGuides** collection to add ruler grid lines to the **RulerGuides** collection. This example creates horizontal and vertical ruler guides every half inch on the first page of the active publication.

```
Sub SetRulerGuides()
    Dim intCount As Integer
    Dim intPos As Integer
    With ActiveDocument.Pages(1).RulerGuides
        For intCount = 1 To 16
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeVertical
        Next intCount
        intPos = 0
        For intCount = 1 To 21
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeHorizontal
        Next intCount
    End With
End Sub
```

Use the **Count** property to return the total number of ruler guides, horizontal and vertical, in the collection. The following example uses the **Count** property to create a loop that deletes each of the ruler guides in the collection.

```
Sub RemoveAllGuides()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).RulerGuides
        For intCount = 1 To .Count
            .Item(1).Delete
        Next intCount
    End With
End Sub
```

Sections Collection

Document └Sections
└Section

A collection of all the **Section** objects in the document.

Using the Sections collection

Use **Sections**.Item(*index*) where *index* is the index number, to return a single **Section** object. The following example sets the number format and the starting number for the first section of the active document.

```
With ActiveDocument.Sections.Item(1)
    .PageNumberFormat = pbPageNumberFormatArabic
    .PageNumberStart = 1
End With
```

Using **Sections**(*index*) where *index* is the index number, will also return a single **Section** object. The following example sets continues the numbering from the previous section for the second section in the active document.

```
ActiveDocument.Sections(2).ContinueNumbersFromPreviousSection=True
```

Use **Sections**.Count to return the number of sections in the publication. The following example display the number of sections in the first open document.

```
MsgBox Documents(1).Sections.Count
```

Use **Sections**.Add(*StartPageIndex*) where *StartPageIndex* is the index number of the page, to reutrn a new section added to a document. A "Permission denied." error will be returned if the page already contains a section head. The following example adds a new section to the second page of the active document.

```
Dim objSection As Section
Set objSection = ActiveDocument.Sections.Add(StartPageIndex:=2)
```

Use **Sections**(*index*).Delete where *index* is the index number, to delete the specified section from the document. A "Permission denied" error will be returned if an attempt is made to delete the first section. The following example deletes all of the sections of the active document except the first one.

Note The iteration is from the last to the first to avoid a "Subscript out of range." error when accessing a deleted section in the **Sections** collection.

```
Dim i As Long
For i = ActiveDocument.Sections.Count To 1 Step -1
    If i = 1 Then Exit For
```

```
    ActiveDocument.Sections(i).Delete
Next i
```

ShapeNodes Collection

Multiple objects └ShapeNodes
  └ShapeNode

A collection of all the **ShapeNode** objects in the specified freeform. Each **ShapeNode** object represents either a node between segments in a freeform or a control point for a curved segment of a freeform. You can create a freeform manually or by using the **BuildFreeform** and **ConvertToShape** methods.

Using the ShapeNodes Collection

Use the **Nodes** property to return a **ShapeNodes** collection. The following example deletes node four in shape three on the active document. For this example to work, shape three must be a freeform with at least four nodes.

```
Sub DeleteShapeNode()
    ActiveDocument.Pages(1).Shapes(3).Nodes.Delete Index:=4
End Sub
```

Use the **Insert** method to create a new node and add it to the **ShapeNodes** collection. The following example adds a smooth node with a curved segment after node four in shape three on the active document. For this example to work, shape three must e a freeform with at least four nodes.

```
Sub AddCurvedSmoothSegment()
    ActiveDocument.Pages(1).Shapes(3).Nodes.Insert _
        Index:=4, SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingSmooth, X1:=210, Y1:=100
End Sub
```

Use **Nodes**(*index*), where *index* is the node index number, to return a single **ShapeNode** object. If node one in shape three on the active document is a corner point, the following example makes it a smooth point. For this example to work, shape three must be a freeform.

```
Sub SetPointType()
    With ActiveDocument.Pages(1).Shapes(3)
        If .Nodes(1).EditingType = msoEditingCorner Then
            .Nodes.SetEditingType Index:=1, EditingType:=msoEditingS
        End If
    End With
End Sub
```

ShapeRange Collection

Multiple objects └ShapeRange
└Multiple objects

Represents a shape range, which is a set of shapes on a document. A shape range can contain as few as one shape or as many as all the shapes in the document. You can include whichever shapes you want— chosen from among all the shapes in the document or all the shapes in the selection— to construct a shape range. For example, you could construct a **ShapeRange** collection that contains the first three shapes in a document, all the selected shapes in a document, or all the freeform shapes in a document.

Note Most operations that you can do with a **Shape** object, you can also do with a **ShapeRange** object that contains only one shape. Some operations, when performed on a **ShapeRange** object that contains more than one shape, will cause an error.

Using the ShapeRange Collection

This section describes how to:

- Return a set of shapes.
- Return a **ShapeRange** object within a selection or range.
- Align, distribute, and group shapes in a **ShapeRange** object.

Return a set of shapes

Use **Shapes.Range**(*index*), where *index* is the index number of the shape or an array that contains index numbers of shapes, to return a **ShapeRange** collection that represents a set of shapes in a publication. You can use Visual Basic's **Array** function to construct an array of index numbers. The following example sets the fill pattern for shapes one through three on the active publication.

```
Sub ChangeFillPattern()
    ActiveDocument.Pages(1).Shapes.Range(Array(1, 2, 3)) _
        .Fill.PresetGradient Style:=msoGradientDiagonalDown, _
        Variant:=1, PresetGradientType:=msoGradientHorizon
End Sub
```

Although you can use the **Range** method to return any number of shapes, it's simpler to use the **Item** method if you want to return only a single member of the collection. For example, **Shapes**(1) is simpler than **Shapes.Range**(1).

Return a ShapeRange object within a selection or range

Use **Selection.ShapeRange**(*index*), where *index* is the index number of the shape, to return a **Shape** object that represents a shape within a selection. The following example selects the first two shapes on the first page of the active publication and then sets the fill for the first shape in the selection.

```
Sub ChangeFillForShapeRange()
    ActiveDocument.Pages(1).Shapes.Range(Array(1, 2)).Select
    Selection.ShapeRange(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
End Sub
```

This example selects all the shapes on the first page of the active publication, then adds and formats text in the second shape in the range.

```
Sub SelectShapesOnPageOne()
    ActiveDocument.Pages(1).Shapes.Range.Select
    With Selection.ShapeRange(2).TextFrame.TextRange
        .Text = "Shape Number 2"
        .ParagraphFormat.Alignment = pbParagraphAlignmentCenter
        .Font.Size = 25
    End With
End Sub
```

Align, distribute, and group shapes in a ShapeRange object

Use the **Align**, **Distribute**, or **ZOrder** method to position a set of shapes relative to each other or relative to the document. This example specifies a shape range and left-aligns and vertically distributes the shapes on the page.

```
Sub AlignDistibuteShapes()
    Dim rngShapes As ShapeRange
    Set rngShapes = ActiveDocument.Pages(1).Shapes.Range

    With rngShapes
        .Align AlignCmd:=msoAlignLefts, RelativeTo:=msoFalse
        .Distribute DistributeCmd:=msoDistributeVertically, Relative
    End With
End Sub
```

Use the **Group**, **Regroup**, or **Ungroup** method to create and work with a single shape formed from a shape range. The **GroupItems** property for a **Shape** object returns the **GroupShapes** object, which represents all the shapes that were grouped to form one shape. This example specifies a shape range and left-aligns and vertically distributes the shapes on the page.

```
Sub GroupShapes()
    Dim rngShapes As ShapeRange
    Set rngShapes = ActiveDocument.Pages(1).Shapes.Range
    rngShapes.Group

    rngShapes(1).Fill.OneColorGradient _
        Style:=msoGradientFromCenter, _
        Variant:=2, Degree:=1
End Sub
```

Shapes Collection

Multiple objects └Shapes
└Shape
└Multiple objects

A collection of **Shape** objects that represent all the shapes on a page of a publication. Each **Shape** object represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, or picture.

Note If you want to work with a subset of the shapes on a document— for example, to do something to only the AutoShapes on the document or to only the selected shapes— you must construct a **ShapeRange** collection that contains the shapes with which you want to work.

Using the Shapes Collection

Use the **Shapes** property to return the **Shapes** collection. The following example selects all the shapes on the first page of the active publication.

```
Sub SelectAllShapes()
    ActiveDocument.Pages(1).Shapes.SelectAll
End Sub
```

Note If you want to do something (like delete or set a property) to all the shapes in a publication at the same time, use the **Range** method to create a **ShapeRange** object that contains all the shapes in the **Shapes** collection, and then apply the appropriate property or method to the **ShapeRange** object.

Use one of the following methods of the **Shapes** collection: **AddCallout**, **AddConnector**, **AddCurve**, **AddLabel**, **AddLine**, **AddOLEObject**, **AddPolyline**, **AddShape**, **AddTextbox**, or **AddTextEffect** to add a shape to a publication and return a **Shape** object that represents the newly created shape. The following example adds a new shape to the active publication.

```
Sub AddNewShape()
    ActiveDocument.Pages(1).Shapes.AddShape Type:=msoShapeFoldedCorn
        Left:=50, Top:=50, Width:=100, Height:=200
End Sub
```

Use **Shapes**(*index*), where *index* is the index number, to return a single **Shape** object. The following example horizontally flips shape one on the first page of the active publication.

```
Sub FlipShape()
    ActiveDocument.Pages(1).Shapes(1).Flip FlipCmd:=msoFlipHorizonta
End Sub
```

Stories Collection

Document └Stories
└Story
└Multiple objects

Represents all the text in a publication.

Using the Stories collection

Use the **Stories** property of a **Document** object to return a **Stories** collection. Use the **Item** method of the **Stories** collection to access individual **Story** objects.

The **Stories** collection enables efficient access to text in a publication. A simple loop through the **Stories** collection can scan all text in text frames or tables without the need to search each shape on every page of a publication.

The **Stories** collection contains one **Story** object for each unlinked text frame, each chain of linked text frames, and each table in a publication. Text in WordArt frames, OLE objects, and pictures are not included in the **Stories** collection.

This example assigns the first story in the active publication to an object variable.

```
Dim stFirst As Story

stFirst = Application.ActiveDocument.Stories(1)
```

TabStops Collection

ParagraphFormat └TabStops
└TabStop

A collection of **TabStop** objects that represent the custom and default tabs for a paragraph or group of paragraphs.

Using the TabStops Collection

Use the **Tabs** property to return the **TabStops** collection. The following example clears all the custom tab stops from the first paragraph in the active publication.

```
Sub ClearAllTabStops()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .ParagraphFormat.Tabs.ClearAll
End Sub
```

The following example adds a tab stop positioned at 2.5 inches to the selected paragraphs and then displays the position of each item in the **TabStops** collection.

```
Sub Tabs()
    Dim intTab As Integer
    Selection.TextRange.ParagraphFormat.Tabs _
        .Add Position:=InchesToPoints(2.5), _
        Alignment:=pbTabAlignmentLeading, Leader:=pbTabLeaderNone
    With Selection.TextRange.ParagraphFormat
        For intTab = 1 To .Tabs.Count
            MsgBox "Position = " & PointsToInches _
               (.Tabs(intTab).Position) & " inches"
            intTab = intTab + 1
        Next intTab
    End With
End Sub
```

Use the **Add** method to add a tab stop. The following example adds two tab stops to the selected paragraphs. The first tab stop is a left-aligned tab with a dotted tab leader positioned at 1 inch (72 points). The second tab stop is centered and is positioned at 2 inches.

```
Sub AddNewTabs()
    With Selection.TextRange.ParagraphFormat.Tabs
        .Add Position:=InchesToPoints(1), _
            Leader:=pbTabLeaderDot, Alignment:=pbTabAlignmentLeading
        .Add Position:=InchesToPoints(2), _
            Leader:=pbTabLeaderNone, Alignment:=pbTabAlignmentCenter
    End With
End Sub
```

Use **Tabs** (*index*), where *index* is the location of the tab stop (in points) or the index number, to return a single **TabStop** object. Tab stops are indexed numerically from left to right along the ruler. The following example removes the first custom tab stop from the first paragraph in the active publication.

```
Sub ClearTabStop()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .ParagraphFormat.Tabs(1).Clear
End Sub
```

The following example changes the second tab in the selection to a right-aligned tab stop.

```
Sub ChangeTabStop()
    Selection.TextRange.ParagraphFormat.Tabs(2) _
        .Alignment = pbTabAlignmentTrailing
End Sub
```

Tags Collection

Multiple objects └Tags
└Tag

A collection of **Tag** objects that represents tags or custom properties applied to a shape, shape range, page, or publication.

Using the Tags Object

Use the **Tags** property to access the **Tags** collection. Use the **Add** method of the **Tags** collection to add a **Tag** object to a shape, shape range, page, or publication. This example adds a tag to each oval shape on the first page of the active publication.

```
Sub AddNewTag()
    Dim shp As Shape
    With ActiveDocument.Pages(1)
        For Each shp In .Shapes
            If InStr(1, shp.Name, "Oval") > 0 Then
                shp.Tags.Add Name:="Shape", Value:="Oval"
            End If
        Next shp
    End With
End Sub
```

Use the **Count** property to determine if a shape, shape range, page, or publication contains one or more **Tag** objects. This example fills all shapes on the first page of the active publication if the shape's first tag has a value of Oval.

```
Sub FormatTaggedShapes()
    Dim shp As Shape
    With ActiveDocument.Pages(1)
        For Each shp In .Shapes
            If shp.Tags.Count > 0 Then
                If shp.Tags(1).Value = "Oval" Then
                    shp.Fill.ForeColor.RGB = RGB(Red:=255, Green:=0,
                End If
            End If
        Next shp
    End With
End Sub
```

TextStyles Collection

Document └TextStyles
└TextStyle
└Multiple objects

A collection of **TextStyle** objects that represent both the built-in and user-defined styles in a document.

Using the TextStyles Collection

Use the **TextStyles** property to return the **TextStyles** collection. The following example creates a table and lists all the styles in the active publication.

```
Sub ListTextStyles()
    Dim sty As TextStyle
    Dim tbl As Table
    Dim intRow As Integer

    With ActiveDocument
        Set tbl = .Pages(1).Shapes.AddTable(NumRows:=.TextStyles.Cou
            NumColumns:=2, Left:=72, Top:=72, Width:=488, Height:=12
        For Each sty In .TextStyles
            intRow = intRow + 1
            With tbl.Rows(intRow)
                .Cells(1).text = sty.Name
                .Cells(2).text = sty.BaseStyle
            End With
        Next sty
    End With
End Sub
```

Use the **Add** method to create a new user-defined style and add it to the **TextStyles** collection. The following example creates a new style and applies it to the paragraph at the insertion point position.

```
Sub ApplyTextStyle()
    Dim styNew As TextStyle
    Dim fntStyle As Font

    'Create a new style
    Set styNew = ActiveDocument.TextStyles.Add(StyleName:="NewStyle"
    Set fntStyle = styNew.Font

    'Format the Font object
    With fntStyle
        .Name = "Tahoma"
        .Size = 20
        .Bold = msoTrue
    End With

    'Apply the Font object formatting to the new style
    styNew.Font = fntStyle
```

```
    'Apply the new style to the selected paragraph
    Selection.TextRange.ParagraphFormat.TextStyle = "NewStyle"
End Sub
```

WebHiddenFields Collection

WebCommandButton └WebHiddenFields

Represents hidden Web fields that allow a Web page to pass non-visible data to the Web server when a Web page is submitted. The **WebHiddenFields** object enables control of all the hidden fields attached to a Submit command button.

Using the WebHiddenFields object

Use the **HiddenFields** property to access hidden Web fields. This example adds a new hidden Web field to a new Submit command button.

```
Sub CreateActionWebButton()
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl _
                (Type:=pbWebControlCommandButton, Left:=150, _
                Top:=150, Width:=75, Height:=36).WebCommandButton
            .ButtonText = "Submit"
            .ButtonType = pbCommandButtonSubmit
            .HiddenFields.Add Name:="User", Value:="PowerUser"
        End With
    End With
End Sub
```

WebListBoxItems Object

WebListBox └WebListBoxItems

Represents the items in a Web list box control.

Using the WebListBoxItems object

Use the **ListBoxItems** property to access the items in a Web list box. Use the **AddItem** method of the **WebListBoxItems** object to add items to a Web list box. This example creates a new Web list box and adds several items to it. Note that when initially created, a Web list box control contains three default items. This example includes a routine that deletes the default list box items before adding new items.

```
Sub CreateWebListBox()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl(Type:=pbWebControlListBox, Left:=100, _
                Top:=150, Width:=300, Height:=72).WebListBox
            .MultiSelect = msoFalse
            With .ListBoxItems
                For intCount = 1 To .Count
                    .Delete (1)
                Next
                .AddItem Item:="Green"
                .AddItem Item:="Purple"
                .AddItem Item:="Red"
                .AddItem Item:="Black"
            End With
        End With
    End With
End Sub
```

WebNavigationBarHyperlinks Object

WebNavigationBarSet
└WebNavigationBarHyperlinks
└Hyperlink
└Multiple objects

The **WebNavigationBarHyperlinks** represents a collection of all the **Hyperlink** objects of the specified **WebNavigationBarSet** object.

Using the WebNavigationBarHyperlinks Object

Use the **Links** property of the **WebNavigationBarSets** object to return a **WebNavigationBarHyperlinks** object. The following example adds a hyperlink to the first **WebNavigationBarSet** of the active document.

```
Dim objWebNavLinks As WebNavigationBarHyperlinks
Set objWebNavLinks = ActiveDocument.WebNavigationBarSets(1).Links
objWebNavLinks.Add Address:="www.microsoft.com",  _
    TextToDisplay:="Microsoft"
```

Use **WebNavigationBarHyperlinks.Count** to return a Long representing the number of hyperlinks in the **WebNavigationBarHyperlinks** collection of the specified **WebNavigationBarSet** object. The following example displays the number of hyperlinks in the first **WebNavigationBarSet** of the active document.

```
MsgBox ActiveDocument.WebNavigationBarSets(1).Links.Count
```

Use **WebNavigationBarHyperlinks**.Item(*index*), where *index* is the index number, to return a specific **Hyperlink** object from the collection. This example displays the displayed text of the first item in the **WebNavigationBarHyperlinks** collection of the first **WebNavigationBarSet** of the active document.

```
MsgBox ActiveDocument.WebNavigationBarSets(1).Links.Item(1).TextToDi
```

WebNavigationBarSets Collection

Document └WebNavigationBarSets
└WebNavigationBarSet
└WebNavigationBarHyperlinks

A collection of all the **WebNavigationBarSet** objects in the current document. Each **WebNavigationBarSet** represents a Web navigation bar set consisting of hyperlinks.

Remarks

By default there are two **WebNavigationBarSet** objects on each Web wizard page; one is text-only and the other is vertical. These objects correspond to the design of the wizard regardless of whether or not a navigation bar is used on the page.

Using the WebNavigationBarSets Collection

Use the **WebNavigationBarSets** property of the current document to return a **WebNavigationBarSet** object. The following example sets an object variable to the **WebNavigationBarSets** collection of the active document.

```
Dim objWebNavBarSets As WebNavigationBarSets
Set objWebNavBarSets = ActiveDocument.WebNavigationBarSets
```

Use **WebNavigationBarSets.Item***(index*), where *index* is the index number, to return a **WebNavigationBarSet** object from the collection. The following example returns the first Web navigation bar set from the **WebNavigationBarSets** collection.

```
Dim objWebNavBarSet As WebNavigationBarSet
Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets.Item(1)
```

The previous example can also be accomplished using **WebNavigationBarSets**(*index*), where *index* is the index number, to return a **WebNavigationBarSet** object.

```
Dim objWebNavBarSet As WebNavigationBarSet
Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets(1)
```

The previous example can also be accomplished using **WebNavigationBarSets**(*index*) where *index* is a string indicating the name of the Web navigation bar set to return.

```
Dim objWebNavBarSet As WebNavigationBarSet
Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets("WebNavBar
```

Use **WebNavigationBarSets.Count** to return the number of Web navigation bar sets in the collection. This example displays the number of Web navigation bar sets in the current document.

```
MsgBox ActiveDocument.WebNavigationBarSets.Count
```

Use **WebNavigationBarSets.AddToEveryPage**(*Left*, *Top*, [*Width*]), where *Left* is the distance from the left of the page to the left edge of the navigation bar, *Top* is the distance form the top of the page to the top edge of the navigation bar, and

Width is the width of the navigaion bar, to add the specified navigation bar to every page. The following example adds the navigation bar named "WebNavBar1" to every page in the current publication.

```
ActiveDocument.WebNavigationBarSets.Item _
    ("WebNavBarSet1").AddToEveryPage _
    Left:=50, Top:=25
```

WizardProperties Collection

Wizard └WizardProperties
└WizardProperty
└WizardValues

Represents the settings available in a publication design or in a Design Gallery object's wizard.

Using the WizardProperties collection

Use the **Properties** property with a **Wizard** object to return a **WizardProperties** collection. The following example reports on the publication design associated with the active publication, displaying its name and current settings.

```
Dim wizTemp As Wizard
Dim wizproTemp As WizardProperty
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp
    Set wizproAll = .Properties
    MsgBox "Publication Design associated with " _
        & "current publication: " .Name
    For Each wizproTemp In wizproAll
        With wizproTemp
            Debug.Print "   Wizard property: " _
                & .Name & " = " & .CurrentValueId
        End With
    Next wizproTemp
End With
```

Note Depending on the language version of Publisher that you are using, you may receive an error when using the above code. If this occurs, you will need to build in error handlers to circumvent the errors. For more information, see **Wizard Object**.

WizardValues Collection

WizardProperty └WizardValues
└WizardValue

Represents the complete set of valid values for a wizard property.

Using the WizardValues collection

Use the **Values** property of the **WizardProperty** object to return a **WizardValues** collection. The following example displays the current value for the first wizard property in the active publication and then lists all the other possible values.

```
Dim valAll As WizardValues
Dim valLoop As WizardValue

With ActiveDocument.Wizard
    Set valAll = .Properties(1).Values

    MsgBox "Wizard: " & .Name & vbLf & _
        "Property: " & .Properties(1).Name & vbLf & _
        "Current value: " & .Properties(1).CurrentValueId

    For Each valLoop In valAll
        MsgBox "Possible value: " & valLoop.ID & " (" & valLoop.Name
    Next valLoop
End With
```

AdvancedPrintOptions Object

Document └AdvancedPrintOptions
└Multiple objects

Represents the advanced print settings for a publication.

Using the AdvancedPrintOptions object

Use the **AdvancedPrintOptions** property of the **Document** object to return an **AdvancedPrintOptions** object. The following example tests to determine if the active publication has been set to print as separations. If it has, it is set to print only plates for the inks actually used in the publication, and to not print plates for any pages where a color is not used.

```
Sub PrintOnlyInksUsed
        With ActiveDocument.AdvancedPrintOptions
                If .PrintMode = pbPrintModeSeparations Then
                        .InksToPrint = pbInksToPrintUsed
                        .PrintBlankPlates = False
                End If
        End With
End Sub
```

Remarks

The properties of the **AdvancedPrintOptions** object correspond to the options available on the tabs of the **Advanced Print Settings** dialog box.

Application Object

Application └Multiple objects

Represents the Microsoft Publisher application. The **Application** object includes properties and methods that return top-level objects. For example, the **ActiveDocument** property returns a **Document** object.

Using the Application object

Use the **Application** property to return the **Application** object. The following example displays the application name.

```
Sub ShowAppName()
    MsgBox Application.Name
End Sub
```

Remarks

When using Visual Basic for Applications in Microsoft Publisher, all of the properties and methods of the **Application** object can be used without the **Application** object qualifier. For example, instead of typing `Application.ActiveDocument.PrintOut`, you can type `ActiveDocument.PrintOut`. Properties and methods that can be used without the **Application** object qualifier are considered "global." To view the global properties and methods in the Object Browser, click **<globals>** at the top of the list in the **Classes** box. When accessing the Publisher object model from a non-Publisher project, all properties and methods must be fully qualified.

BorderArt Object

BorderArts └BorderArt

Represents an available type of BorderArt. BorderArt is picture borders that can be applied to text boxes, picture frames, or rectangles. The **BorderArt** object is a member of the **BorderArts** collection. The **BorderArts** collection contains all BorderArt available for use in the specified publication.

Using the BorderArt Object

Use the **Item** property of a **BorderArts** collection to return a specific BorderArt object. The *Index* argument of the **Item** property can be the number or name of the BorderArt object.

This example returns the BorderArt "Apples" from the active publication.

```
Dim bdaTemp As BorderArt

Set bdaTemp = ActiveDocument.BorderArts.Item (Index:="Apples")
```

Use the **Name** property to specify which type of BorderArt you want applied to a picture. The following example sets all the BorderArt in a document to the same type using the **Name** property.

```
Sub SetBorderArtByName()

Dim anyPage As Page
Dim anyShape As Shape
Dim strBorderArtName As String

strBorderArtName = Document.BorderArts(1).Name

        For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Name = strBorderArtName
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

Note Because **Name** is the default property of both the **BorderArt** object and the **BorderArtFormat** object, you do not need to state it explicitly when setting the BorderArt type. The statement `Shape.BorderArtFormat = Document.BorderArts(1)` is equivalent to `Shape.BorderArtFormat.Name = Document.BorderArts(1).Name`.

Remarks

The **BorderArts** collection includes any custom BorderArt types created by the user for the specified publication.

BorderArtFormat Object

Shape └BorderArtFormat
└ColorFormat

Represents the formatting of the BorderArt applied to the specified shape.

Using the BorderArtFormat Object

Use the **BorderArt** property of a shape to return a **BorderArtFormat** object.

The following example returns the BorderArt of the first shape on the first page of the active publication, and displays the name of the BorderArt in a message box.

```
Dim bdaTemp As BorderArtFormat

Set bdaTemp = ActiveDocument.Pages(1).Shapes(1).BorderArt
MsgBox "BorderArt name is: " &bdaTemp.Name
```

Use the **Set** method to specify which type of BorderArt you want applied to a picture. The following example tests for the existence of BorderArt on each shape for each page of the active document. Any BorderArt found is set to the same type.

```
Sub SetBorderArt()
Dim anyPage As Page
Dim anyShape As Shape
Dim strBorderArtName As String

strBorderArtName = Document.BorderArts(1).Name

        For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Set(strBorderArtName)
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

You can also use the **Name** property to specify which type of BorderArt you want applied to a picture. The following example sets all the BorderArt in a document to the same type using the **Name** property.

```
Sub SetBorderArtByName()
Dim anyPage As Page
Dim anyShape As Shape
```

```
Dim strBorderArtName As String

        strBorderArtName = Document.BorderArts(1).Name

        For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Name = strBorderArtName
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

Note Because **Name** is the default property of both the **BorderArt** and **BorderArtFormat** objects, you do not need to state it explicitly when setting the BorderArt type. The statement `Shape.BorderArtFormat = Document.BorderArts(1)` is equivalent to `Shape.BorderArtFormat.Name = Document.BorderArts(1).Name`.

Use the **Delete** method to remove BorderArt from a picture. The following example tests for the existence of border art on each shape for each page of the active document. If border art exists, it is deleted.

```
Sub DeleteBorderArt()
        Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Delete
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

Remarks

BorderArt are picture borders that can be applied to text boxes, picture frames, or rectangles.

CalloutFormat Object

Multiple objects └CalloutFormat

Contains properties and methods that apply to line callouts.

Using the CalloutFormat object

Use the **Callout** property to return a **CalloutFormat** object. The following example adds a callout to the active publication, adds text to the callout, then specifies the following attributes for the callout:

- a vertical accent bar that separates the text from the callout line (**Accent** property)
- the angle between the callout line and the side of the callout text box will be 30 degrees (**Angle** property)
- there will be no border around the callout text (**Border** property)
- the callout line will be attached to the top of the callout text box (**PresetDrop** method)
- the callout line will contain three segments (**Type** property)

```
Sub AddFormatCallout()
    With ActiveDocument.Pages(1).Shapes.AddCallout(Type:=msoCalloutO
            Left:=150, Top:=150, Width:=200, Height:=100)
        With .TextFrame.TextRange
            .Text = "This is a callout."
            With .Font
                .Name = "Stencil"
                .Bold = msoTrue
                .Size = 30
            End With
        End With
        With .Callout
            .Accent = MsoTrue
            .Angle = msoCalloutAngle30
            .Border = MsoFalse
            .PresetDrop msoCalloutDropTop
            .Type = msoCalloutThree
        End With
    End With
End Sub
```

Cell Object

CellRange └Cell
└Multiple objects

Represents a single table cell. The **Cell** object is a member of the **CellRange** collection. The **CellRange** collection represents all the cells in the specified object.

Using the Cell object

Use **Cells**(*index*), where *index* is the cell number, to return a **Cell** object. This example merges the first two cells of the first column of the specified table.

```
Sub MergeCell()
    With ActiveDocument.Pages(1).Shapes(2).Table.Columns(1)
        .Cells(1).Merge MergeTo:=.Cells(2)
    End With
End Sub
```

This example applies a thick border around the first cell in the second column of the specified table.

```
Sub OutlineBorderCell()
    With ActiveDocument.Pages(1).Shapes(2).Table.Columns(2).Cells(1)
        .BorderLeft.Weight = 5
        .BorderRight.Weight = 5
        .BorderTop.Weight = 5
        .BorderBottom.Weight = 5
    End With
End Sub
```

CellBorder Object

Cell └CellBorder
└ColorFormat

Represents the color and weight settings for cell borders.

Using the CellBorder object

Use the various border properties of the **Cell** object to return the different borders of a cell (left, right, top, bottom, and diagonal). The following example retrieves the top border of the first cell in a table.

```
Dim cbTemp As CellBorder

Set cbTemp = ActiveDocument.Pages(1) _
    .Shapes(1).Table.Cells.Item(1).BorderTop
```

Use the **Color** and **Weight** properties of the **CellBorder** object to format the appearance of a cell border. The following example makes the left border of the first cell in a table red and two points thick.

```
Dim cbTemp As CellBorder

Set cbTemp = ActiveDocument.Pages(1) _
    .Shapes(1).Table.Cells.Item(1).BorderLeft

cbTemp.Color.RGB = RGB(255, 0, 0)
cbTemp.Weight = 2
```


Show All

ColorCMYK Object

ColorFormat └ColorCMYK

Represents a cyan-magenta-yellow-black (CMYK) color value.

Using the ColorCMYK object

Use the **CMYK** property of a **ColorFormat** object to return a **ColorCMYK** object. Use the **Cyan**, **Magenta**, **Yellow**, and **Black** properties of the **ColorCMYK** object to individually set each of the four colors in the CMYK color value. Use the **SetCMYK** method on a **ColorCMYK** object to set all four colors at once.

The following example retrieves the CMYK color value of shape one's fill and changes it to another CMYK color value.

```
Dim cmykColor As ColorCMYK

Set cmykColor =
ActiveDocument.Pages(1).Shapes(1).Fill.ForeColor.CMYK

cmykColor.SetCMYK Cyan:=0, Magenta:=255, Yellow:=255, Black:=50
```

ColorFormat Object

Multiple objects └ColorFormat
└ColorCMYK

Represents the color of a one-color object or the foreground or background color of an object with a gradient or patterned fill. You can set colors to an explicit red-green-blue value by using the **RGB** property.

Using the ColorFormat object

Use one of the properties listed in the following table to return a **ColorFormat** object.

Use this property	With this object	To return a ColorFormat object that represents this
BackColor	**FillFormat**	Background fill color (used in a shaded or patterned fill)
ForeColor	**FillFormat**	Foreground fill color (or simply the fill color for a solid fill)
BackColor	**LineFormat**	Background line color (used in a patterned line)
ForeColor	**LineFormat**	Foreground line color (or just the line color for a solid line)
ForeColor	**ShadowFormat**	Shadow color
ExtrusionColor	**ThreeDFormat**	Color of the sides of an extruded object

Use the **RGB** property to set a color to an explicit red-green-blue value. The following example adds a rectangle to the active publication and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
Sub GradientFill()
    With ActiveDocument.Pages(1).Shapes _
            .AddShape(Type:=msoShapeRectangle, _
            Left:=90, Top:=90, Width:=90, Height:=50).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .BackColor.RGB = RGB(170, 170, 170)
        .TwoColorGradient msoGradientHorizontal, 1
    End With
End Sub
```

ColorScheme Object

Multiple objects └ColorScheme
└ColorFormat

Represents a color scheme, which is a set of eight colors used for the different elements of a publication. Each color is represented by a **ColorFormat** object. The **ColorScheme** object is a member of the **ColorSchemes** collection. The **ColorSchemes** collection contains all the color schemes available to Microsoft Publisher.

Using the ColorScheme Object

Use the **ColorScheme** property of a **Document** object to return the color scheme for the current publication. The following example sets the fill value of three shapes on the first page to the return value (in RGB format) of three of the eight **ColorScheme** colors.

```
Sub ReturnColorsAndApplyToShapes()
    Dim lngAccent1 As Long
    Dim lngAccent2 As Long
    Dim lngAccent3 As Long

    With ActiveDocument
        With .ColorScheme
            lngAccent1 = .Colors(pbSchemeColorAccent1).RGB
            lngAccent2 = .Colors(pbSchemeColorAccent2).RGB
            lngAccent3 = .Colors(pbSchemeColorAccent3).RGB
        End With
        With .Pages(1)
            .Shapes(1).Fill.ForeColor.RGB = lngAccent1
            .Shapes(2).Fill.ForeColor.RGB = lngAccent2
            .Shapes(3).Fill.ForeColor.RGB = lngAccent3
        End With
    End With

End Sub
```

Use the **Name** property to return a color scheme name. The following example lists in a text box all the color schemes available to Publisher.

```
Sub ListColorShemes()

    Dim clrScheme As ColorScheme
    Dim strSchemes As String

    For Each clrScheme In Application.ColorSchemes
        strSchemes = strSchemes & clrScheme.Name & vbLf
    Next
    ActiveDocument.Pages(1).Shapes.AddTextbox( _
        Orientation:=pbTextOrientationHorizontal, _
        Left:=72, Top:=72, Width:=400, Height:=500).TextFrame _
        .TextRange.Text = strSchemes

End Sub
```

Column Object

Columns └column
└CellRange

Represents a single table column. The **Column** object is a member of the **Columns** collection. The **Columns** collection includes all the columns in a table, selection, or range.

Using the Column object

Use **Columns**(*index*), where *index* is the column number, to return a single **Column** object. The index number represents the position of the column in the **Columns** collection (counting from left to right). This example selects column three in the first shape in the active publication. This example assumes the first shape is a table and not another type of shape.

```
Sub SelectColumn()
    ActiveDocument.Pages(2).Shapes(1).Table.Columns(3).Cells.Select
End Sub
```

Use the **Item** method of a **Columns** collection to return a **Column** object. This example enters text into the first cell of the third column of the specified table and formats the text with a bold, 15-point font. This example assumes the first shape is a table and not another type of shape.

```
Sub ColumnHeading()
    With ActiveDocument.Pages(2).Shapes(1).Table.Columns(3) _
            .Cells(1).Text
        .Text = "Sales"
        .Font.Bold = msoTrue
        .Font.Size = 15
    End With
End Sub
```

Use the **Delete** method to delete a column from a table. This example deletes the column added in the above example.

```
Sub DeleteColumn()
    ActiveDocument.Pages(2).Shapes(1).Table.Columns(3).Delete
End Sub
```

ConnectorFormat Object

Multiple objects └ConnectorFormat
└Shape

Contains properties and methods that apply to connectors. A connector is a line that attaches two other shapes at points called connection sites. If you rearrange shapes that are connected, the geometry of the connector will be automatically adjusted so that the shapes remain connected.

Using the ConnectorFormat object

Use the **ConnectorFormat** property of the **Shape** object or the **ShapeRange** collection to return a **ConnectorFormat** object. Use the **BeginConnect** and **EndConnect** methods of the **ConnectorFormat** object to attach the ends of the connector to other shapes in the publication. Use the **RerouteConnections** method of the **Shape** object and **ShapeRange** collection to automatically find the shortest path between the two shapes connected by the connector. Use the **Connector** property to see whether a shape is a connector.

Note that you assign a size and a position when you add a connector to the **Shapes** collection, but the size and position are automatically adjusted when you attach the beginning and end of the connector to other shapes in the collection. Therefore, if you intend to attach a connector to other shapes, the initial size and position you specify are irrelevant. Likewise, you specify which connection sites on a shape to attach the connector to when you attach the connector, but using the **RerouteConnections** method after the connector is attached may change which connection sites the connector attaches to, making your original choice of connection sites irrelevant.

The following example adds two rectangles to the active publication and connects them with a curved connector.

```
Dim shpAll As Shapes
Dim firstRect As Shape
Dim secondRect As Shape

Set shpAll = ActiveDocument.Pages(1).Shapes
Set firstRect = shpAll.AddShape(Type:=msoShapeRectangle, _
    Left:=100, Top:=50, Width:=200, Height:=100)
Set secondRect = shpAll.AddShape(Type:=msoShapeRectangle, _
    Left:=300, Top:=300, Width:=200, Height:=100)


With shpAll.AddConnector(Type:=msoConnectorCurve, BeginX:=0, _
    BeginY:=0, EndX:=0, EndY:=0).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    .Parent.RerouteConnections
End With
```

Document Object

Multiple objects └Document
└Multiple objects

Represents a publication.

Using ActiveDocument

Use the **ActiveDocument** property to refer to the current publication. This example adds a table to the first page of the active publication.

```
Sub NewTable()
    With ActiveDocument.Pages(1).Shapes
        .AddTable NumRows:=3, NumColumns:=3, Left:=72, Top:=300, _
            Width:=488, Height:=36
        With .Item(1).Table.Rows(1)
            .Cells(1).TextRange.Text = "Column1"
            .Cells(2).TextRange.Text = "Column2"
            .Cells(3).TextRange.Text = "Column3"
        End With
    End With
End Sub
```

You can also write the above routine by using a reference to the **ThisDocument** module. This example uses a **ThisDocument** reference instead of **ActiveDocument**.

```
Sub PrintPublication()
    With ThisDocument.Pages(1).Shapes
        .AddTable NumRows:=3, NumColumns:=3, Left:=72, Top:=300, _
            Width:=488, Height:=36
        With .Item(1).Table.Rows(1)
            .Cells(1).TextRange.Text = "Column1"
            .Cells(2).TextRange.Text = "Column2"
            .Cells(3).TextRange.Text = "Column3"
        End With
    End With
End Sub
```

DropCap Object

TextRange └DropCap
└ColorFormat

Represents a dropped capital letter at the beginning of a paragraph.

Using the DropCap Object

Use the **DropCap** property to return a **DropCap** object. The following example sets a dropped capital letter for the first letter of each paragraph in the first shape on the first page of the active publication. This example assumes that the specified shape is a text box and not another type of shape.

```
Sub ApplyDropCap()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .DropCap.ApplyCustomDropCap Size:=3, Span:=3, Bold:=True
End Sub
```

Field Object

TextRange └Fields
└Field
└Multiple objects

Represents a field. The **Field** object is a member of the **Fields** collection. The **Fields** collection represents the fields in a selection, range, or publication.

Using the Field Object

Use **Fields** (*index*), where *index* is the index number, to return a single **Field** object. The index number represents the position of the field in the selection, range, or publication. The following counts the number of fields in the active publication and displays the count in a message.

```
Sub CountFields()
    Dim pagPage As Page
    Dim shpShape As Shape
    Dim fldField As Field
    Dim intFields As Integer
    Dim intCount As Integer

    For Each pagPage In ActiveDocument.Pages
        For Each shpShape In pagPage.Shapes
            If shpShape.Type = pbTextFrame Then
                intCount = intCount + shpShape.TextFrame.TextRange.F
            End If
        Next
    Next
    If intCount > 0 Then
        MsgBox "You have " & intCount & " fields in your publication
    Else
        MsgBox "You have no fields in your publication."
    End If
End Sub
```

The **pbFieldPageNumber** constant is a member of the **PbFieldType** group of constants, which includes all the various field types.

FillFormat Object

Multiple objects └FillFormat
    └ColorFormat

Represents fill formatting for a shape. A shape can have a solid, gradient, texture, pattern, picture, or semitransparent fill.

Using the FillFormat object

Use the **Fill** property to return a **FillFormat** object. The following example adds a shape to the active document and then sets the gradient and color for the shape's fill.

```
Sub AddShapeAndSetFill()
    With ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShapeHeart
            Left:=90, Top:=90, Width:=90, Height:=80).Fill
        .ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)
        .OneColorGradient Style:=msoGradientHorizontal, _
            Variant:=1, Degree:=1
    End With
End Sub
```

Remarks

Many of the properties of the **FillFormat** object are read-only. To set one of these properties, you have to apply the corresponding method.

FindReplace Object

Multiple objects └FindReplace
└TextRange

Represents the criteria for a find operation. The properties and methods of the **FindReplace** object correspond to the options in the **Find and Replace** dialog box.

Using the FindReplace Object

Use the **Find** property to return a **FindReplace** object. The following example selects the next occurrence of the word "factory".

```
With ActiveDocument.Find
    .Clear
    .FindText = "factory"
    .Execute
End With
```

Set the **ReplaceScope** property to determine the extent of the search. The following example replaces the first occurrence of the name "Visual Basic Scripting Edition" with "VBScript".

```
With ActiveDocument.Find
    .Clear
    .FindText = "Visual Basic Scripting Edition"
    .ReplaceWithText = "VBScript"
    .ReplaceScope = pbReplaceScopeOne
    .Execute
End With
```

Remarks

When the **ReplaceScope** property is set to **pbReplaceScopeOne** or **pbReplaceScopeAll**, the **ReplaceWithText** property must be set to avoid the text from being replaced with the default value of an empty **String** for that property.

Examples

The following example illustrates how the font attributes of the FoundTextRange can be accessed when **ReplaceScope** is set to **pbReplaceScopeNone**.

```
Dim objFindReplace As FindReplace

Set objFindReplace = ActiveDocument.Find
With objFindReplace
    .Clear
    .FindText = "important"
    .ReplaceScope = pbReplaceScopeNone
    Do While .Execute = True
        If .FoundTextRange.Font.Italic = msoFalse Then
            .FoundTextRange.Font.Italic = msoTrue
        End If
    Loop
End With
```

Font Object

Multiple objects └Font
└ColorFormat

Contains font attributes (font name, font size, color, and so on) for an object.

Using the Font Object

Use the **Font** property to return the **Font** object. The following instruction applies bold formatting to the selection.

```
Sub BoldText()
    Selection.TextRange.Font.Bold = True
End Sub
```

The following example formats the first paragraph in the active publication as 24-point Arial and italic.

```
Sub FormatText()
    Dim txtRange As TextRange
    Set txtRange = ActiveDocument.Pages(1).Shapes(1).TextFrame.TextR
    With txtRange.Font
        .Bold = True
        .Name = "Arial"
        .Size = 24
    End With
End Sub
```

The following example changes the formatting of the Heading 2 style in the active publication to Arial and bold.

```
Sub FormatStyle()
    With ActiveDocument.TextStyles("Normal").Font
        .Name = "Tahoma"
        .Italic = True
        .Size = 15
    End With
End Sub
```

You can also duplicate a **Font** object by using the **Duplicate** property. The following example creates a new character style with the character formatting from the selection as well as italic formatting. The formatting of the selection isn't changed.

```
Sub DuplicateFont()
    Dim fntNew As Font
```

```
    Set fntNew = Selection.TextRange.Font.Duplicate
    fntNew.Italic = True
    ActiveDocument.TextStyles.Add(StyleName:="Italics").Font = fntNe
End Sub
```

FreeformBuilder Object

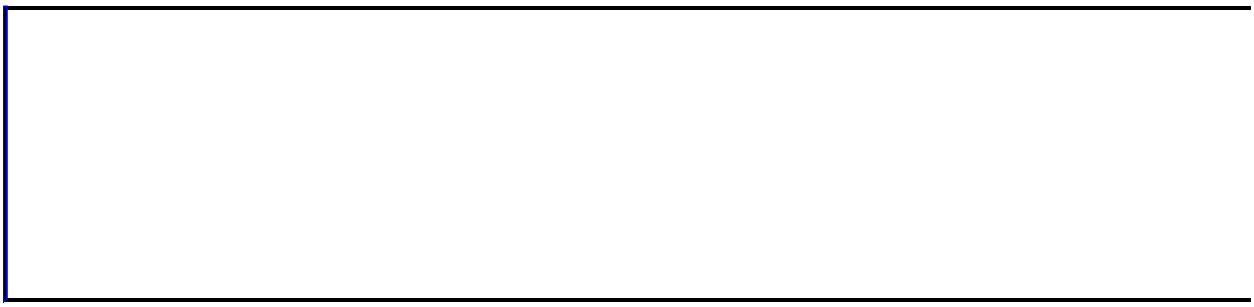
FreeformBuilder

Represents the geometry of a freeform while it's being built.

Using the FreeformBuilder Object

Use the **BuildFreeform** method of the **Shapes** collection to return a **FreeformBuilder** object. Use the **AddNodes** method to add nodes to the freeform. Use the **ConvertToShape** method to create the shape defined in the **FreeformBuilder** object and add it to the **Shapes** collection. The following example adds a freeform with four segments to the active document.

```
Sub CreateNewFreeFormShape()
    With ActiveDocument.Pages(1).Shapes.BuildFreeform( _
            EditingType:=msoEditingCorner, X1:=360, Y1:=200)
        .AddNodes SegmentType:=msoSegmentCurve, _
            EditingType:=msoEditingCorner, X1:=380, Y1:=230, _
            X2:=400, Y2:=250, X3:=450, Y3:=300
        .AddNodes SegmentType:=msoSegmentCurve, _
            EditingType:=msoEditingAuto, X1:=480, Y1:=200
        .AddNodes SegmentType:=msoSegmentLine, _
            EditingType:=msoEditingAuto, X1:=480, Y1:=400
        .AddNodes SegmentType:=msoSegmentLine, _
            EditingType:=msoEditingAuto, X1:=360, Y1:=200
        .ConvertToShape
    End With
End Sub
```



HeaderFooter Object

Page └HeaderFooter
  └TextRange

Represents the header or footer of a master page.

Using the HeaderFooter Object

Use **MasterPages.Header** or **MasterPages.Footer** to return a **HeaderFooter** object. The following example adds text to the header of the first master page of the active document.

```
Dim objHeader As HeaderFooter
Set objHeader = ActiveDocument.MasterPages(1).Header
objHeader.TextRange.Text = "Master Page 1 Header"
```

Use **HeaderFooter.Delete** to delete any existing content from a header or footer. Calling this method does not delete the text frame, just the contents of it. The following example deletes all of the header and footer content of all the master pages in a publication.

```
Dim objMasterPage As page
For Each objMasterPage In ActiveDocument.masterPages
    objMasterPage.Header.Delete
    objMasterPage.Footer.Delete
Next
```

Use **HeaderFooter.TextRange** to return a **TextRange** object representing the header or footer of a master page. Any header or footer content manipulation is done with through this property of the **HeaderFooter** object. The following example first deletes any existing content and then adds some boilerplate text to the header of a master page.

```
Dim objHeader As HeaderFooter
Set objHeader = ActiveDocument.MasterPages(1).Header
With objHeader
    .Delete
    .TextRange.Text = "<Insert Address Here>"
End With
```

Hyperlink Object

Multiple objects └Hyperlink
    └Multiple objects

Represents a hyperlink. The **Hyperlink** object is a member of the **Hyperlinks** collection and the **Shape** and **ShapeRange** objects.

Using the Hyperlink Object

Use the **Hyperlink** property to return a **Hyperlink** object associated with a shape (a shape can have only one hyperlink). The following example deletes the hyperlink associated with the first shape in the active document.

```
Sub DeleteHyperlink()
    ActiveDocument.Pages(1).Shapes(1).Hyperlink.Delete
End Sub
```

Use **Hyperlinks**(*index*), where *index* is the index number, to return a single **Hyperlink** object from a document, range, or selection. The following example deletes the first hyperlink in the selection.

```
Sub DeleteSelectedHyperlink()
    If Selection.TextRange.Hyperlinks.Count >= 1 Then
        Selection.TextRange.Hyperlinks(1).Delete
    End If
End Sub
```

Use the **Add** method to add a hyperlink. The following example adds a hyperlink to the selected text.

```
Sub AddHyperlinkToSelectedText()
    Selection.TextRange.Hyperlinks.Add Text:=Selection.TextRange, _
        Address:="http://www.tailspintoys.com/"
End Sub
```

Use the **Address** property to add or change the address to a hyperlink. The following example adds a shape to the active publication and then adds a hyperlink to the shape.

```
Sub AddHyperlinkToShape()
    With ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=200, Width:=300, Height:=300)
        .Hyperlink.Address = "http://www.tailspintoys.com/"
    End With
End Sub
```

Label Object

Multiple objects └Label

Represents a single unique label design available on the system.

Using the Label object

Use the **Label** property to return the **Label** object.

Each label design available on the system resides in the **AvailableLabels** collection, which is accessed by using the **AvailableLabels** property on the **PageSetup** object.

The following properties of the **Label** object are read/write when the **Label** object is returned using **.PageSetup.Label**. These properties are read-only if the **Label** object is returned using any other method.

- **TopMargin**
- **LeftMargin**
- **HorizontalGap**
- **VerticalGap**

The following example uses the **Label** property to return the fifth label available on the system, and then some of the label's properties are set.

```
With ActiveDocument.PageSetup
    .Label = .AvailableLabels(5)  ' Label 5 is Avery 5164
    Set theLabel = .Label
    With theLabel
        .LeftMargin = InchesToPoints(0.15)
        .TopMargin = InchesToPoints(0.15)
        .HorizontalGap = InchesToPoints(0.1)
        .VerticalGap = InchesToPoints(0.1)
    End With
End With
```

LayoutGuides Object

Multiple objects └LayoutGuides

Represents the measurement grid that appears superimposed on publication pages as an aid to laying out design elements.

Using the LayoutGuides object

Use the **LayoutGuides** property of the **Document** object to return a **LayoutGuides** object. Use the **LayoutGuide** object's margin properties and **Rows** and **Columns** properties to set how many rows and columns are displayed in the layout guides and where they appear on a page.

This example sets the margins of the active presentation to two inches.

```
With ActiveDocument.LayoutGuides
    .MarginTop = Application.InchesToPoints(Value:=2)
    .MarginBottom = Application.InchesToPoints(Value:=2)
    .MarginLeft = Application.InchesToPoints(Value:=2)
    .MarginRight = Application.InchesToPoints(Value:=2)
End With
```

LineFormat Object

Multiple objects └LineFormat
└ColorFormat

Represents line and arrowhead formatting. For a line, the **LineFormat** object contains formatting information for the line itself; for a shape with a border, this object contains formatting information for the shape's border.

Using the LineFormat Object

Use the **Line** property to return a **LineFormat** object. The following example adds a blue, dashed line to the active document. There's a short, narrow oval at the line's starting point and a long, wide triangle at its end point.

```
Sub FormatLine()
    With ActiveDocument.Pages(1).Shapes.AddLine(BeginX:=100, _
            BeginY:=100, EndX:=200, EndY:=300).Line
         .DashStyle = msoLineDashDotDot
         .ForeColor.RGB = RGB(50, 0, 128)
         .BeginArrowheadLength = msoArrowheadShort
         .BeginArrowheadStyle = msoArrowheadOval
         .BeginArrowheadWidth = msoArrowheadNarrow
         .EndArrowheadLength = msoArrowheadLong
         .EndArrowheadStyle = msoArrowheadTriangle
         .EndArrowheadWidth = msoArrowheadWide
    End With
End Sub
```

LinkFormat Object

Multiple objects └LinkFormat

Represents the linking characteristics for an OLE object or picture.

Using the LinkFormat Object

Use the **LinkFormat** property for a shape or field to return a **LinkFormat** object. The following example updates the links to all linked OLE objects on the first page of the active publication.

```
Sub FindOLEObjects()
    Dim shpShape As Shape

    For Each shpShape In ActiveDocument.Pages(1).Shapes
        If shpShape.Type = pbLinkedOLEObject Then
            shpShape.LinkFormat.Update
        End If
    Next shpShape
End Sub
```

Remarks

Not all types of shapes and fields can be linked to a source. Use the **Type** property for the **Shape** object to determine whether a particular shape can be linked.

Use the **Update** method to update links. To return or set the full path for a particular link's source file, use the **SourceFullName** property.


Show All

MailMerge Object

Document └MailMerge
└MailMergeDataSource

Represents the mail merge and catalog merge functionality in Publisher.

Using the MailMerge Object

Use the **MailMerge** property to return the **MailMerge** object. The **MailMerge** object is always available regardless of whether the mail merge or catalog merge operation has begun. The following example merges and prints the main publication with the first three data records in the attached data source.

```
Sub SelectiveMerge()
    Dim mrgMain As MailMerge
    Set mrgMain = ActiveDocument.MailMerge
    With mrgMain.DataSource
        .FirstRecord = 1
        .LastRecord = 3
    End With
    mrgMain.Execute True
End Sub
```


Show All

MailMergeDataField Object

MailMergeDataFields └MailMergeDataField

Represents a single merge field in a data source. The **MailMergeDataField** object is a member of the **MailMergeDataFields** collection. The **MailMergeDataFields** collection includes all the data fields in a mail merge or catalog merge data source (for example, Name, Address, and City).

Using the MailMergeDataField Object

Use **DataFields** (*index*), where *index* is the data field name or index number, to return a single **MailMergeDataField** object. The index number represents the position of the data field in the mail merge data source. This example retrieves the name of the first field and value of the first record of the FirstName field in the data source attached to the active publication.

```
Sub GetDataFromSource()
    With ActiveDocument.MailMerge.DataSource
        MsgBox "Field Name: " & .DataFields.Item(1).Name & _
            "Value: " & .DataFields.Item("FirstName").Value
    End With
End Sub
```

Remarks

You cannot add fields to the **MailMergeDataFields** collection. All data fields in a data source are automatically included in the **MailMergeDataFields** collection.


Show All

MailMergeDataSource Object

MailMerge └MailMergeDataSource
└Multiple objects

Represents the data source in a mail merge or catalog merge operation.

Using the MailMergeDataSource Object

Use the **DataSource** property to return the **MailMergeDataSource** object. The following example displays the name of the data source associated with the active publication.

```
Sub ShowDataSourceName()
    If ActiveDocument.MailMerge.DataSource.Name <> "" Then _
        MsgBox ActiveDocument.MailMerge.DataSource.Name
End Sub
```

The following example tests the open data source associated with the active publication to determine whether the LastName field includes the name Fuller.

```
Sub FindSelectedRecord()
    With ActiveDocument.MailMerge
        If .DataSource.FindRecord(FindText:="Fuller", _
                Field:="LastName") = True Then
            MsgBox "Data was found"
        End If
    End With
End Sub
```


Show All

MailMergeFilterCriterion Object

MailMergeFilterCriterion

Represents a filter to be applied to an attached mail merge or catalog merge data source. The **MailMergeFilterCriterion** object is a member of the **MailMergeFilters** object.

Using the MailMergeFilterCriterion object

Each filter is a line in a query string. Use the **Column**, **Comparison**, **CompareTo**, and **Conjunction** properties to return or set the data source query criterion. The following example changes an existing filter to remove from the mail merge all records that do not have a Region field equal to "WA". This example assumes that a data source is attached to the active publication.

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next
    End With
End Sub
```

Use the **Add** method of the **MailMergeFilters** object to add a new filter criterion to the query. This example adds a new line to the query string and then applies the combined filter to the data source. This example assumes that a data source is attached to the active publication.

```
Sub FilterDataSource()
    With ActiveDocument.MailMerge.DataSource
        .Filters.Add Column:="Region", _
            Comparison:=msoFilterComparisonIsBlank, _
            Conjunction:=msoFilterConjunctionAnd
        .ApplyFilter
    End With
End Sub
```

MailMergeMappedDataField Object

MailMergeMappedDataFields └MailMergeMappedDataField

Represents a single mapped data field. The **MailMergeMappedDataField** object is a member of the **MailMergeMappedDataFields** collection.

A mapped data field is a field contained within Publisher that represents commonly used name or address information, such as First Name. If a data source contains a First Name field or a variation (such as First_Name, FirstName, First, or FName), the field in the data source will automatically map to the corresponding mapped data field. If a publication is to be merged with more than one data source, mapped data fields make it unnecessary to reenter the fields into the publication to agree with the field names in the database.

Using the MailMergeMappedDataField object

Use **MappedDataFields**(*index*) to return a **MailMergeMappedDataField** object. This example returns the data source field name for the **pbFirstName** mapped data field. This example assumes the current publication is a mail merge publication. A blank string value returned for the **DataFieldName** property indicates that the mapped data field is not mapped to a field in the data source.

```
Sub MappedFieldName()
    Dim strMappedDataField As String
    With ActiveDocument.MailMerge.DataSource
        strMappedDataField = .MappedDataFields(pbFirstName).DataFiel
        If strMappedDataField <> "" Then
            MsgBox "The mapped data field 'FirstName' is mapped to "
            & .MappedDataFields(pbFirstName).DataFieldName & "."
        Else
            MsgBox "The mapped data field 'FirstName' is not " & _
                "mapped to any of the data fields in your " & _
                "data source."
        End If
    End With
End Sub
```

OLEFormat Object

Multiple objects └OLEFormat
└ObjectVerbs

Represents the OLE characteristics, other than linking (see the **LinkFormat** object), for an OLE object, ActiveX control, or field.

Using the OLEFormat Object

Use the **OLEFormat** property for a shape or field to return an **OLEFormat** object. The following example activates all OLE objects in the active publication.

```
Sub ActivateOLEObjects()
    Dim shpShape As Shape

    For Each shpShape In ActiveDocument.Pages(1).Shapes
        If shpShape.Type = pbLinkedOLEObject Then
            shpShape.OLEFormat.Activate
        End If
    Next
End Sub
```

Remarks

Not all types of shapes and fields have OLE capabilities. Use the **Type** property for the **Shape** object to determine into which category the specified shape falls.

Use the **Activate** and **DoVerb** methods to automate an OLE object.

Options Object

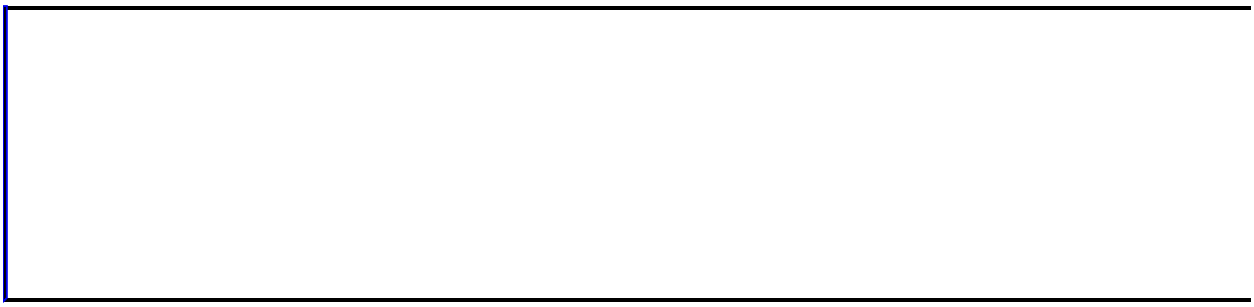
Application └Options

Represents application and publication options in Microsoft Publisher. Many of the properties for the **Options** object correspond to items in the **Options** dialog box (**Tools** menu).

Using the Options Object

Use the **Options** property to return the **Options** object. The following example sets four application options for Publisher.

```
Sub SetSpecialOptions()
    With Options
        .AllowBackgroundSave = True
        .DragAndDropText = True
        .AutoHyphenate = True
        .MeasurementUnit = pbUnitInch
    End With
End Sub
```



Page Object

Multiple objects └Page
└Multiple objects

Represents a page in a publication. The **Pages** collection contains all the **Page** objects in a publication.

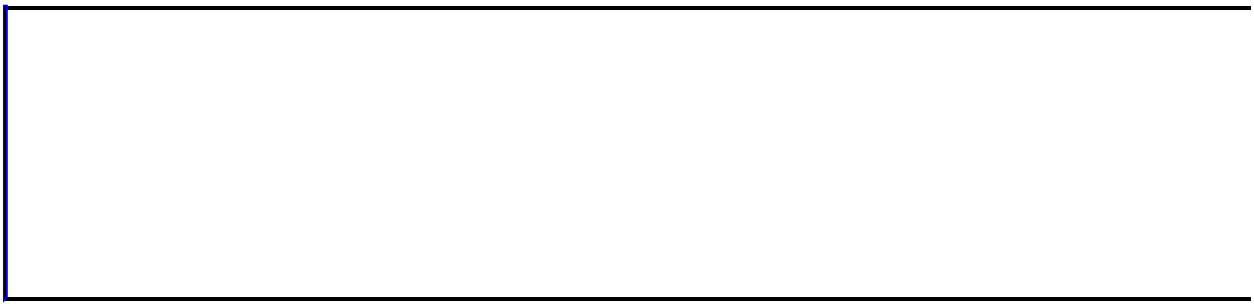
Using the Page object

Use **Pages**(*index*) to return a single **Page** object. The following example adds new text to the first shape on the first page in the active publication.

```
Sub AddPageNumberField()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange
        .InsertAfter " This text is added after the existing text."
        .Font.Size = 15
    End With
End Sub
```

Use the **FindBypageID** property to locate a **Page** object using the application assigned page ID. Use the **Add** method to create a new page and add it to the publication. The following example adds a new page to the active publication and then looks for that page using the page ID.

```
Sub FindPage()
    Dim lngPageID As Long

    'Get page ID
    lngPageID = ActiveDocument.Pages.Add(Count:=1, After:=1).PageID

    'Use page ID to add a new shape to the page
    ActiveDocument.Pages.FindByPageID(PageID:=lngPageID) _
        .Shapes.AddShape Type:=msoShape5pointStar, _
        Left:=200, Top:=72, Width:=50, Height:=50

End Sub
```



PageBackground Object

Page └PageBackground
└FillFormat

Represents the background of a page.

Using the PageBackground Object

Use the **Background** property of a **Page** object to return a **PageBackground** object. The following example creates a **PageBackground** object and sets it to the background of the first page of the active document.

```
Dim objPageBackground As PageBackground
Set objPageBackground = ActiveDocument.Pages(1).Background
```

Use **PageBackground**.Exists to determine if a background already exists for the specified **Page** object. The following example builds upon the previous example. First a **PageBackground** object is created and set to the background of the first page of the active document. Then a test is made to check if a background exists for the page already. If not then one is created by calling the **Create** method of the **PageBackground** object.

```
Dim objPageBackground As PageBackground
Set objPageBackground = ActiveDocument.Pages(1).Background
If objPageBackground.Exists = False Then
    objPageBackground.Create
End If
```

Use **PageBackground**.Fill to return a **FillFormat** object. The following example builds upon the previous example. First a **PageBackground** object is created and set to the background of the first page of the active document. Then a test is made to check if a background exists for the page already. If not then one is created by calling the **Create** method of the **PageBackground** object. A **FillFormat** object is returned by using the **Fill** property of the **PageBackground** object. A few of the available properties of the **FillFormat** object are then set.

```
Dim objPageBackground As PageBackground
Dim objFillFormat As FillFormat

Set objPageBackground = ActiveDocument.Pages(1).Background
If objPageBackground.Exists = False Then
    objPageBackground.Create
End If

Set objFillFormat = objPageBackground.Fill
With objFillFormat
```

```
    .BackColor.RGB = RGB(Red:=0, GReen:=155, Blue:=99)
    .ForeColor.RGB = RGB(Red:=155, GReen:=234, Blue:=0)
    .TwoColorGradient msoGradientDiagonalDown, 4
End With
```

Use **PageBackground**.Delete to delete a background for the specified page. The following example deletes the background of the first page in the active document.

```
ActiveDocument.Pages(1).Background.Delete
```

PageSetup Object

Document └PageSetup
└Multiple objects

Contains information about the page setup for the pages in a publication.

Using the PageSetup object

Use the **PageSetup** property to return the **PageSetup** object. The following example sets all pages in the active publication to be 8.5 inches wide and 11 inches high.

```
Sub SetPageSetupOptions()
    With ActiveDocument.PageSetup
        .PageHeight = 11 * 72
        .PageWidth = 8.5 * 72
    End With
End Sub
```

ParagraphFormat Object

Multiple objects └ParagraphFormat
└TabStops

Represents all the formatting for a paragraph.

Using the ParagraphFormat Object

Use the **ParagraphFormat** property to return the **ParagraphFormat** object for a paragraph or paragraphs. The **ParagraphFormat** property returns the **ParagraphFormat** object for a selection, range, or style. The following example centers the paragraph at the cursor position. This example assumes that the first shape is a text box and not another type of shape.

```
Sub CenterParagraph()
    Selection.TextRange.ParagraphFormat _
        .Alignment = pbParagraphAlignmentCenter
End Sub
```

Use the **Duplicate** property to copy an existing **ParagraphFormat** object. The following example duplicates the paragraph formatting of the first paragraph in the active publication and stores the formatting in a variable. This example duplicates an existing **ParagraphFormat** object and then changes the left indent to one inch, creates a new textbox, inserts text into it, and applies the paragraph formatting of the duplicated paragraph format to the text.

```
Sub DuplicateParagraphFormating()
    Dim pfmtDup As ParagraphFormat

    Set pfmtDup = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.ParagraphFormat.Duplicate

    pfmtDup.LeftIndent = Application.InchesToPoints(1)

    With ActiveDocument.Pages.Add(Count:=1, After:=1)
        With .Shapes.AddTextbox(pbTextOrientationHorizontal, _
            Left:=72, Top:=72, Width:=200, Height:=100)
            With .TextFrame.TextRange
                .Text = "This is a test of how to use " & _
                    "the ParagraphFormat object."
                .ParagraphFormat = pfmtDup
            End With
        End With
    End With

End Sub
```

PhoneticGuide Object

Field └PhoneticGuide

Represents base text with supplementary text appearing above it as a guide to pronunciation.

Using the PhoneticGuide object

Use the **PhoneticGuide** property of a **Field** object to return an existing **PhoneticGuide** object. Use the **AddPhoneticGuide** method of a **Fields** collection to create a new **PhoneticGuide** object.

The following example adds a new **PhoneticGuide** object to the active publication.

```
Selection.TextRange.Fields.AddPhoneticGuide _
     Range:=Selection.TextRange, Text:="ver-E nIs", _
     Alignment:=pbPhoneticGuideAlignmentCenter, _
     Raise:=11, FontSize:=7
```

PictureFormat Object

Multiple objects └PictureFormat

Contains properties and methods that apply to pictures.

Using the PictureFormat Object

Use the **PictureFormat** property to return a **PictureFormat** object. The following example sets the brightness, contrast, and color transformation for shape one on the active document and crops 18 points off the bottom of the shape. For this example to work, shape one must be either a picture or an OLE object.

```
Sub FormatPicture()
    With ActiveDocument.Pages(1).Shapes(1).PictureFormat
        .Brightness = 0.6
        .Contrast = 0.7
        .ColorType = msoPictureGrayscale
        .CropBottom = 18
    End With
End Sub
```

Plate Object

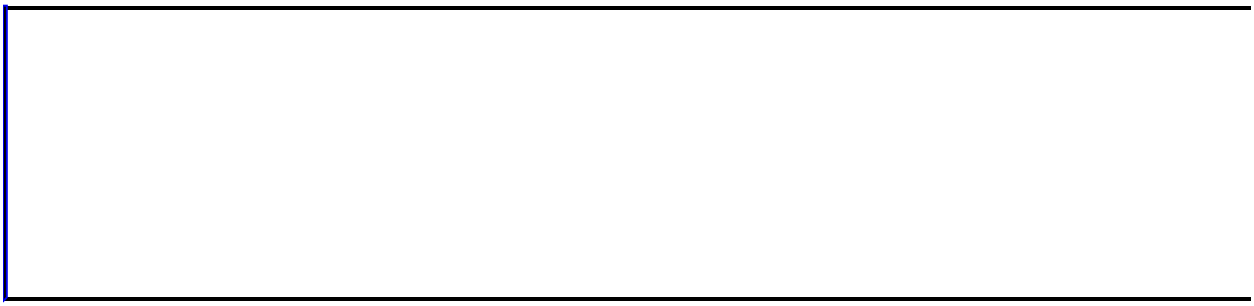
Document └Plates
└Plate
└ColorFormat

Represents a single printer's plate. The **Plate** object is a member of the **Plates** collection.

Using the Plate object

Use the **Add** method of the **Plates** collection to create a new plate. This example creates a new spot-color plate collection and adds a plate to it.

```
Sub AddNewPlates()
    Dim plts As Plates
    Set plts = ActiveDocument.CreatePlateCollection(Mode:=pbColorMod
    plts.Add
    With plts(1)
        .Color.RGB = RGB(Red:=255, Green:=0, Blue:=0)
        .Luminance = 4
    End With
End Sub
```



PrintablePlate Object

AdvancedPrintOptions └PrintablePlates
└PrintablePlate

Represents a single plate to be printed for the publication. The **PrintablePlate** object is a member of the **PrintablePlates** collection.

Using the PrintablePlate object

Use the **FindPlateByInkName** method of the **PrintablePlates** collection to return a specific plate by referencing its ink name. The following example returns a spot color plate and sets several of its properties. The example assumes that separations have been specified as the active publication's print mode.

```
Sub SetPlatePropertiesByInkName()

Dim pplPlate As PrintablePlate
ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

    Set pplPlate = ActiveDocument.AdvancedPrintOptions.PrintablePlat

    With pplPlate
        .Angle = 75
        .Frequency = 133
        .PrintPlate = True
    End With

End Sub
```

The following example returns a list of the printable plates currently in the collection for the active publication. The example assumes that separations have been specified as the active publication's print mode.

```
Sub ListPrintablePlates()
    Dim pplTemp As PrintablePlates
    Dim pplLoop As PrintablePlate


    Set pplTemp = ActiveDocument.AdvancedPrintOptions.PrintablePlate
    Debug.Print "There are " & pplTemp.Count & " printable plates in

    For Each pplLoop In pplTemp
        With pplLoop
            Debug.Print "Printable Plate Name: " & .Name
            Debug.Print "Index: " & .Index
            Debug.Print "Ink Name: " & .InkName
            Debug.Print "Plate Angle: " & .Angle
            Debug.Print "Plate Frequency: " & .Frequency
            Debug.Print "Print Plate?: " & .PrintPlate
        End With
    Next pplLoop
End Sub
```

Remarks

To specify custom frequency or angle settings for a printable plate, the **UseCustomHalftone** of the **AdvancedPrintOptions** object must be set to **True**.

The **PrintablePlates** collection is generated when a publication's print mode is set to separations. Returns "Permission Denied" when any other print mode is specified.

The **PrintablePlates** collection represents the plates that will actually be printed for the publication, based on:

- The plates (if any) you have defined for the publication
- The advanced print options specified

You cannot programmatically create a printable plates collection, or add a printable plate to the collection.

Use the **PrintMode** property of the **AdvancedPrintOptions** object to set the publication to print as separations.

Each **PrintablePlate** object corresponds to a plate listed on the **Separations** tab of the **Advanced Print Settings** dialog box.

PrintableRect Object

AdvancedPrintOptions └PrintableRect

Represents the sheet area within which the specified printer will print. The printable rectangle is determined by the printer based on the sheet size specified. The printable rectangle of the printer sheet should not be confused with the area within the margins of the publication page; it may be larger or smaller than the publication page.

Using the PrintableRect object

Use the **PrintableRect** property of the **AdvancedPrintOptions** object to return a **PrintableRect** object. The following example returns printable rectangle boundaries for the printer sheet of the active publication.

```
Sub ListPrintableRectBoundaries()

With ActiveDocument.AdvancedPrintOptions.PrintableRect

    Debug.Print "Printable area is " & _
            PointsToInches(.Width) & _
            " by " & PointsToInches(.Height) & " inches."
    Debug.Print "Left Boundary: " & PointsToInches(.Left) & _
                    " inches (from left)."
    Debug.Print "Right Boundary: " & PointsToInches(.Left + .Width)
                    " inches (from left)."
    Debug.Print "Top Boundary: " & PointsToInches(.Top) & _
                    " inches(from top)."
    Debug.Print "Bottom Boundary: " & PointsToInches(.Top + .Height)
                    " inches(from top)."

End With

End Sub
```

Remarks

In cases in which the printer sheet and the publication page size are identical, the publication page is centered on the printer sheet and none of the printer's marks print, even if they are selected.

ReaderSpread Object

Page └ReaderSpread
  └Page

Represents the reader spread (not the printer spread) for the page. A reader spread generally contains one or two pages. The **ReaderSpread** object properties provide information about whether pages are facing and how those pages are laid out. For example, in facing page view, pages two and three can be side-by-side or one on top of the other.

Using the ReaderSpread object

Use the **ReaderSpread** property to access the **ReaderSpread** object for a page. Use the **PageCount** property to determine if the reader spread includes one page or two facing pages. This example checks to see if the reader spread includes less than two pages. If it does, it changes the reader spread to include two pages.

```
Sub SetFacingPages()
    With ActiveDocument
        If .Pages.Count >= 2 Then
            If .Pages(2).ReaderSpread.PageCount < 2 Then _
                .ViewTwoPageSpread = True
        End If
    End With
End Sub
```

Row Object

Rows
└Row
└CellRange

Represents a row in a table. The **Row** object is a member of the **Rows** collection. The **Rows** collection includes all the rows in a specified table.

Using the Row object

Use **Rows**(*index*), where *index* is the row number, to return a single **Row** object. The index number represents the position of the row in the **Rows** collection (counting from left to right). This example selects the first row in the first shape on the second of the active publication. This example assumes the specified shape is a table and not another type of shape.

```
Sub SelectRow()
    ActiveDocument.Pages(2).Shapes(1).Table.Rows(1).Cells.Select
End Sub
```

Use the **Item** method of a **Rows** collection to return a **Row** object. This example sets the fill for all even numbered rows and clears the fill for all odd numbered rows in the specified table. This example assumes the specified shape is a table and not another type of shape.

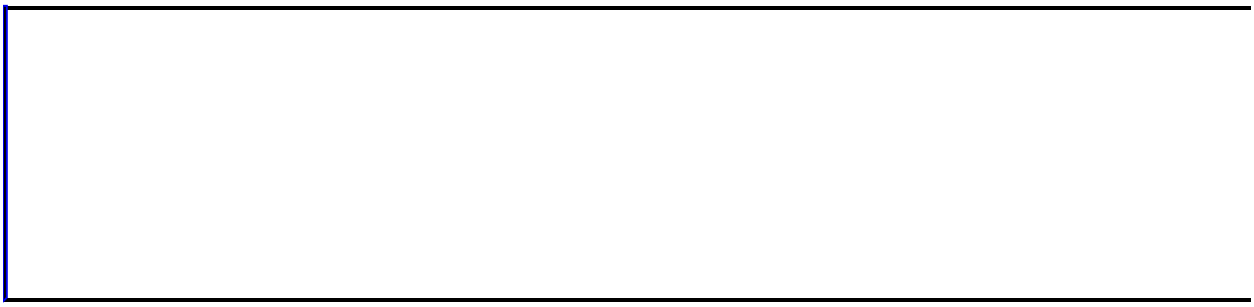
```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            If celTable.Row Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
        Next celTable
    Next rowTable

End Sub
```

Use the **Add** method to add a row to a table. This example adds a row to the specified table on the second page of the active publication, and then adjusts the width, merges the cells, and sets the fill color. This example assumes the first shape is a table and not another type of shape.

```
Sub NewRow()
    Dim rowNew As Row

    Set rowNew = ActiveDocument.Pages(2).Shapes(1).Table.Rows _
        .Add(BeforeRow:=3)
    With rowNew
        .Height = 2
        .Cells.Merge
        .Cells(1).Fill.ForeColor.RGB = RGB(Red:=0, Green:=0, Blue:=0
    End With
End Sub
```

Use the **Delete** method to delete a row from a table. This example deletes the row added in the above example.

```
Sub DeleteRow()
    ActiveDocument.Pages(2).Shapes(1).Table.Rows(3).Delete
End Sub
```



RulerGuide Object

Page └RulerGuides
└RulerGuide

Represents a grid line used to align objects on a page. The **RulerGuide** object is a member of the **RulerGuides** collection.

Using the RulerGuide object

Use the **Add** method of the **RulerGuides** collection to create a new ruler grid line. Use the **Item** property to reference a ruler guide. Use the **Position** property to change the position of a grid line, and use the **Delete** method to remove a grid line. This example creates a new ruler guide, moves it, and then deletes it.

```
Sub AddChangeDeleteGuide()
    Dim rgLine As RulerGuide
    With ActiveDocument.Pages(1).RulerGuides
        .Add Position:=InchesToPoints(1), _
            Type:=pbRulerGuideTypeVertical

        MsgBox "The ruler guide position is at one inch."

        .Item(1).Position = InchesToPoints(3)
        MsgBox "The ruler guide is now at three inches."

        .Item(1).Delete
        MsgBox "The ruler guide has been deleted."
    End With
End Sub
```

ScratchArea Object

Document └ScratchArea
└Shapes

Represents the area outside the boundaries of publication pages where layout elements may be stored with no effect on publication output.

Using the ScratchArea object

Use the **ScratchArea** property of the **Document** object to return a scratch area. Use the **Shapes** property of the **ScratchArea** object to return the collection of shapes that are currently on a scratch area.

This example assigns the first shape on the scratch area of the active document to a variable.

```
Dim saPage As ScratchArea
Dim objFirst As Object

saPage = Application.ActiveDocument.ScratchArea
objFirst = saPage.Shapes(1)
```

Section Object

Document └Sections
└Section

Represents a Section of a publication or document.

Using the Section Object

Use **Sections**.Item(*index*) where *index* is the index number, to return a single **Section** object. The following example sets a **Section** object to the first section in the **Sections** collection of the active document.

```
Dim objSection As Section
Set objSection = ActiveDocument.Sections.Item(1)
```

Use **Sections**.Add(*StartPageIndex*) where *StartPageIndex* is the index number of the page, to return a new section added to a document. A "Permission denied." error will be returned if the page already contains a section head. The following example adds a Section object to the second page of the active document.

```
Dim objSection As Section
Set objSection = ActiveDocument.Sections.Add(StartPageIndex:=2)
```

Selection Object

Multiple objects └Selection
└Multiple objects

Represents the current selection in a window or pane. A selection represents either a selected (or highlighted) area in the publication, or it represents the insertion point if nothing in the publication is selected. There can only be one **Selection** object per publication window pane, and only one **Selection** object in the entire application can be active.

Using the Selection Object

Use the **Selection** property to return the **Selection** object. If no object qualifier is used with the **Selection** property, Publisher returns the selection from the active pane of the active publication window. The following example copies the current selection from the active publication.

```
Sub CopySelection()
    Selection.ShapeRange.Copy
End Sub
```

The following example determines what type of item is selected and if it is an autoshape, fills the first shape in the selection with color. This example assumes there is at least one item selected in the active pubication.

```
Sub SelectedShape()
    If Selection.Type = pbSelectionShape Then
        Selection.ShapeRange.Item(1).Fill.ForeColor _
            .RGB = RGB(Red:=200, Green:=20, Blue:=255)
    End If
End Sub
```

The following example copies the selection and pastes it into the first shape on the second page of the active publication.

```
Sub CopyPasteSelection()
    Selection.TextRange.Copy
    With ActiveDocument.Pages(2).Shapes(1).TextFrame.TextRange
        .Collapse Direction:=pbCollapseEnd
        .InsertAfter NewText:=vbLf
        .Paste
    End With
End Sub
```

ShadowFormat Object

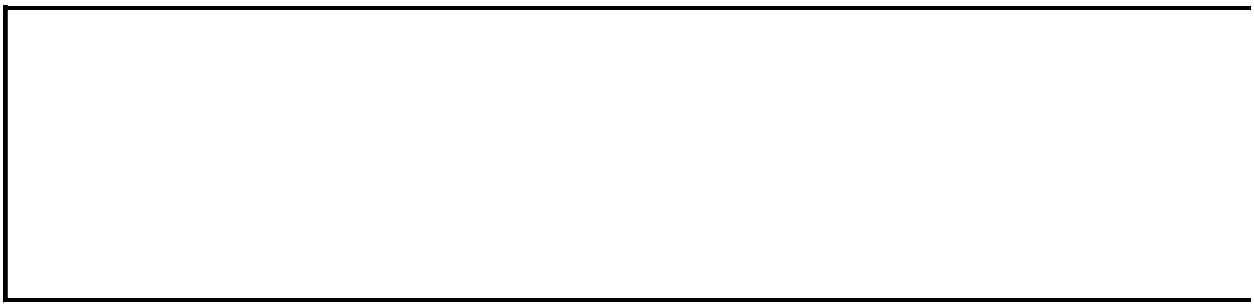
Multiple objects └ShadowFormat
└ColorFormat

Represents shadow formatting for a shape.

Using the ShadowFormat Object

Use the **Shadow** property to return a **ShadowFormat** object. The following example adds a shadowed rectangle to the active document. The pink shadow is offset 7 points to the right of the rectangle and 7 points above it.

```
Sub FormatShadow()
    With ActiveDocument.Pages(1).Shapes.AddShape( _
            Type:=msoShapeRectangle, Left:=72, Top:=72, _
            Width:=100, Height:=200).Shadow
        .ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=150)
        .Obscured = msoTrue
        .OffsetX = 7
        .OffsetY = -7
        .Visible = True
    End With
End Sub
```



Shape Object

Multiple objects └Shape
└Multiple objects

Represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, ActiveX control, or picture. The **Shape** object is a member of the **Shapes** collection, which includes all the shapes on a page or in a selection.

Note There are three objects that represent shapes: the **Shapes** collection, which represents all the shapes on a document; the **ShapeRange** collection, which represents a specified subset of the shapes on a document (for example, a **ShapeRange** object could represent shapes one and four on the document, or it could represent all the selected shapes on the document); the **Shape** object, which represents a single shape on a document. If you want to work with several shape at the same time or with shapes within the selection, use a **ShapeRange** collection.

Using the Shape Object

This section describes how to:

- Return an existing shape on a document.
- Return a shape or shapes within a selection.
- Return a newly created shape.
- Work with a group of shapes.
- Format a shape.
- Use other important shape properties.

Return an existing shape on a document

Use **Shapes** (*index*), where *index* is the name or the index number, to return a single **Shape** object. The following example horizontally flips shape one on the active document.

```
Sub FlipShape()
    ActiveDocument.Pages(1).Shapes(1).Flip FlipCmd:=msoFlipHorizonta
End Sub
```

The following example horizontally flips the shape named "Rectangle 1" on the active document.

```
Sub FlipShapeByName()
    ActiveDocument.Pages(1).Shapes("Rectangle 1") _
        .Flip FlipCmd:=msoFlipHorizontal
End Sub
```

Each shape is assigned a default name when it is created. For example, if you add three different shapes to a document, they might be named "Rectangle 2," "TextBox 3," and "Oval 4." To give a shape a more meaningful name, set the **Name** property of the shape.

Return a shape or shapes within a selection

Use **Selection.ShapeRange**(*index*), where *index* is the name or the index number, to return a **Shape** object that represents a shape within a selection. The following example sets the fill for the first shape in the selection, assuming that the selection contains at least one shape.

```
Sub FillSelectedShape()
    Selection.ShapeRange(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
End Sub
```

The following example sets the fill for all the shapes in the selection, assuming that the selection contains at least one shape.

```
Sub FillAllSelectedShapes()
    Dim shpShape As Shape
    For Each
shpShape In Selection.ShapeRange

shpShape.Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)
    Next shpShape
End Sub
```

Return a newly created shape

To add a **Shape** object to the collection of shapes for the specified document and return a **Shape** object that represents the newly created shape, use one of the following methods of the **Shapes** collection: **AddCallout**, **AddConnector**, **AddCurve**, **AddLabel**, **AddLine**, **AddOLEObject**, **AddPolyline**, **AddShape**, **AddTextBox** or **AddTextEffect**. The following example adds a rectangle to the active document.

```
Sub AddNewShape()
    ActiveDocument.Pages(1).Shapes.AddShape Type:=msoShapeRectangle,
        Left:=400, Top:=72, Width:=100, Height:=200
End Sub
```

Work with a group of shapes

Use **GroupItems** (*index*), where *index* is the shape name or the index number within the group, to return a **Shape** object that represents a single shape in a grouped shape. Use the **Group** or **Regroup** method to group a range of shapes and return a single **Shape** object that represents the newly formed group. After a group has been formed, you can work with the group the same way you work with any other shape. This example adds three shapes to the active publication, groups the shapes, and sets the fill color for each of the shapes in the group

```
Sub WorkWithGroupShapes()

    With ActiveDocument.Pages(1).Shapes
        .AddShape Type:=msoShapeIsoscelesTriangle, Left:=100, _
            Top:=72, Width:=100, Height:=100
        .AddShape Type:=msoShapeIsoscelesTriangle, Left:=250, _
            Top:=72, Width:=100, Height:=100
        .AddShape Type:=msoShapeIsoscelesTriangle, Left:=400, _
            Top:=72, Width:=100, Height:=100
        .SelectAll

        With Selection.ShapeRange
            .Group
            .GroupItems(1).Fill.ForeColor _
                .RGB = RGB(Red:=255, Green:=0, Blue:=0)
            .GroupItems(2).Fill.ForeColor _
                .RGB = RGB(Red:=0, Green:=255, Blue:=0)
            .GroupItems(3).Fill.ForeColor _
                .RGB = RGB(Red:=0, Green:=0, Blue:=255)
        End With
    End With
End Sub
```

Format a shape

Use the **Fill** property to return the **FillFormat** object, which contains all the properties and methods for formatting the fill of a closed shape. The **Shadow** property returns the **ShadowFormat** object, which you use to format a shadow. Use the **Line** property to return a **LineFormat** object, which contains properties and methods for formatting lines and arrows. The **TextEffect** property returns the **TextEffectFormat** object, which you use to format WordArt. The **Callout** property returns the **CalloutFormat** object, which you use to format line callouts. The **TextWrap** property returns the **WrapFormat** object, which you use to define how text wraps around shapes. The **ThreeD** property returns the **ThreeDFormat** object, which you use to create 3-D shapes. You can use the **PickUp** and **Apply** methods to transfer formatting from one shape to another.

Use the **SetShapesDefaultProperties** method for a **Shape** object to set the formatting for the default shape for the document. New shapes inherit many of their attributes from the default shape.

Use other important shape properties

Use the **Type** property to specify the type of shape: freeform, AutoShape, OLE object, callout, or linked picture, for instance. Use the **AutoShapeType** property to specify the type of AutoShape: oval, rectangle, or balloon, for instance.

Use the **Width** and **Height** properties to specify the size of the shape.

Use **TextFrame** and **TextRange** properties to return the **TextFrame** and **TextRange** objects, respectively, which contain all the properties and methods for inserting and formatting text within shapes and publications and linking the text frames together. The following example adds a text box to the first page of the active publication, then adds text to it and formats the text.

```
Sub CreateNewTextBox()
    With ActiveDocument.Pages(1).Shapes.AddTextbox( _
        Orientation:=pbTextOrientationHorizontal, Left:=100, _
        Top:=100, Width:=200, Height:=100).TextFrame.TextRange
        .Text = "This is a textbox."
        With .Font
            .Name = "Stencil"
            .Bold = msoTrue
            .Size = 30
        End With
    End With
End Sub
```

ShapeNode Object

ShapeNodes └ShapeNode

Represents the geometry and the geometry-editing properties of the nodes in a user-defined freeform. Nodes include the vertices between the segments of the freeform and the control points for curved segments. The **ShapeNode** object is a member of the **ShapeNodes** collection. The **ShapeNodes** collection contains all the nodes in a freeform.

Using the ShapeNode Object

Use **Nodes**(*index*), where *index* is the node index number, to return a single **ShapeNode** object. If node one in shape three on the active document is a corner point, the following example makes it a smooth point. For this example to work, shape one must be a freeform.

```
Sub ChangeNodeType()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Nodes(1).EditingType = msoEditingCorner Then
            .Nodes.SetEditingType Index:=1, EditingType:=msoEditingS
        End If
    End With
End Sub
```

Story Object

Multiple objects └Story
    └Multiple objects

Represents the text in an unlinked text frame, text flowing between linked text frames, or text in a table cell. The **Story** object is a member of the **TextFrame** and **TextRange** objects and the **Stories** collection.

Using the Story object

Use the **Story** property to return the **Story** object in a text range or text frame. This example returns the story in the selected text range and, if it is in a text frame, inserts text into the text range.

```
Sub AddTextToStory()
    With Selection.TextRange.Story
        If .HasTextFrame Then .TextRange _
            .InsertAfter NewText:=vbLf & "This is a test."
    End With
End Sub
```

Use **Stories**(*index*), where *index* is the number of the story, to return an individual **Story** object. This example determines if the first story in the active publication has a text frame and, if it does, formats the paragraphs in the story with a half inch first line indent and a six-point spacing before each paragraph.

```
Sub StoryParagraphFirstLineIndent()
    With ActiveDocument.Stories(1)
        If .HasTextFrame Then
            With .TextFrame.TextRange.ParagraphFormat
                .FirstLineIndent = InchesToPoints(0.5)
                .SpaceBefore = 6
            End With
        End If
    End With
End Sub
```

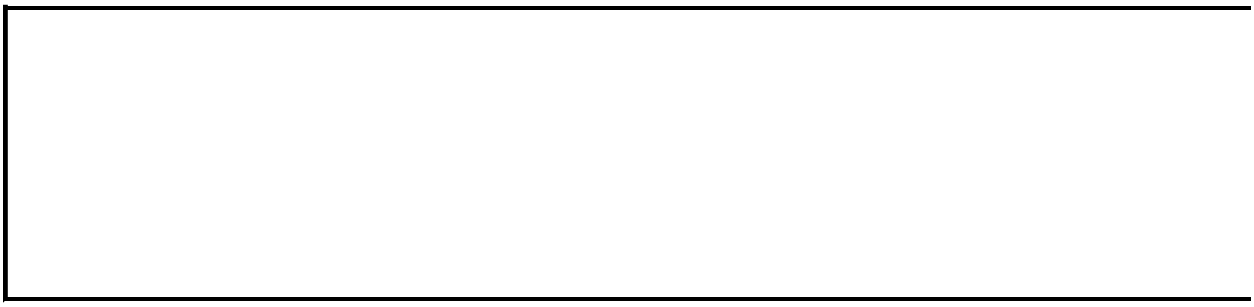


Table Object

Multiple objects └[Table](#)
└Multiple objects

Represents a single table.

Using the Table Object

Use the **Table** property to return a **Table** object. The following example selects the specified table in the active publication.

```
Sub SelectTable()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Type = pbTable Then _
            .Table.Cells.Select
    End With
End Sub
```

Use the **AddTable** method to add a **Shape** object representing a table at the specified range. The following example adds a 5x5 table on the first page of the active publication, and then selects the first column of the new table.

```
Sub NewTable()
    With ActiveDocument.Pages(1).Shapes.AddTable(NumRows:=5, NumColu
        Left:=72, Top:=300, Width:=400, Height:=100)
        .Table.Columns(1).Cells.Select
    End With
End Sub
```

TabStop Object

TabStops └TabStop

Represents a single tab stop. The **TabStop** object is a member of the **TabStops** collection. The **TabStops** collection represents all the custom and default tab stops in a paragraph or group of paragraphs.

Using the TabStop object

Use **Tabs** (***index***), **where *index* is the location of the tab stop (in points) or the index number, to return a single TabStop object. Tab stops are indexed numerically from left to right along the ruler. The following example removes the first custom tab stop from the selected paragraphs.**

```
Sub ClearTabStop()
    Selection.TextRange.ParagraphFormat.Tabs(1).Clear
End Sub
```

The following example adds a right-aligned tab stop positioned at 2 inches to the selected paragraphs.

```
Sub ChangeTabStop()
    Selection.TextRange.ParagraphFormat.Tabs(2) _
        .Alignment = pbTabAlignmentTrailing
End Sub
```

Use the **Add** method to add a tab stop. The following example adds two tab stops to the selected paragraphs. The first tab stop is a left-aligned tab with a dotted tab leader positioned at 1 inch (72 points). The second tab stop is centered and is positioned at 2 inches.

```
Sub AddNewTabs()
    With Selection.TextRange.ParagraphFormat.Tabs
        .Add Position:=InchesToPoints(1), _
            Leader:=pbTabLeaderDot, Alignment:=pbTabAlignmentLeading
        .Add Position:=InchesToPoints(2), _
            Leader:=pbTabLeaderNone, Alignment:=pbTabAlignmentCenter
    End With
End Sub
```

Remarks

Set the **DefaultTabStop** property to adjust the spacing of default tab stops.

Tag Object

Tags └Tag

Represents a tag or a custom property that you can create for a shape, shape range, page, or publication. Each **Tag** object contains the name of a custom property and a value for that property. **Tag** objects are members of the **Tags** collection.

Create a tag when you want to be able to selectively work with specific members of a collection, based on an attribute that isn't already represented by a built-in property.

Using the Tag object

Use the **Item** method of the **Tags** collection to return a **Tag** object. This example fills all shapes on the first page of the active publication if the shape's first tag has a value of Oval.

```
Sub FormatTaggedShapes()
    Dim shp As Shape
    With ActiveDocument.Pages(1)
        For Each shp In .Shapes
            If shp.Tags.Count > 0 Then
                If shp.Tags.Item(1).Value = "Oval" Then
                    shp.Fill.ForeColor.RGB = RGB(Red:=255, Green:=0,
                End If
            End If
        Next
    End With
End Sub
```

Use the **Add** method to add a Tag object. This example adds a tag to all oval shapes in the active publication.

```
Sub TagShapes()
    Dim shp As Shape
    With ActiveDocument.Pages(1)
        For Each shp In .Shapes
            If InStr(1, shp.Name, "Oval") > 0 Then
                shp.Tags.Add Name:="Oval", Value:="This is an oval s
            End If
        Next shp
    End With
End Sub
```

TextEffectFormat Object

Multiple objects └TextEffectFormat

Contains properties and methods that apply to WordArt objects.

Using the TextEffectFormat Object

Use the **TextEffect** property to return a **TextEffectFormat** object. The following example sets the font name and formatting for shape one on the first page of the active publication. For this example to work, shape one must be a WordArt object.

```
Sub FormatWordArt()
    With ActiveDocument.Pages(1).Shapes(1).TextEffect
        .FontName = "Courier New"
        .FontBold = MsoTrue
        .FontItalic = MsoTrue
    End With
End Sub
```


Show All

TextFrame Object

Multiple objects └TextFrame
    └Multiple objects

Represents the text frame in a **Shape** object. Contains the text in the text frame as well as the properties that control the margins and orientation of the text frame.

Using the TextFrame Object

Use the **TextFrame** property to return the **TextFrame** object for a shape. The **TextRange** property returns a **TextRange** object that represents the range of text inside the specified text frame. The following example adds text to the text frame of shape one in the active publication, and then formats the new text.

```
Sub AddTextToTextFrame()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange
        .Text = "My Text"
        With .Font
            .Bold = msoTrue
            .Size = 25
            .Name = "Arial"
        End With
    End With
End Sub
```

Note Some shapes don't support attached text (lines, freeforms, pictures, and OLE objects, for example). If you attempt to return or set properties that control text in a text frame for those objects, an error occurs.

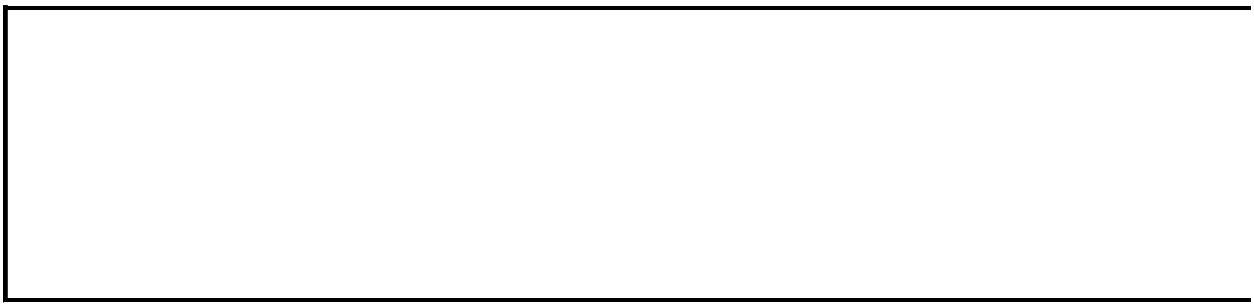
Use the **HasTextFrame** property to determine whether the shape has a text frame and the **HasText** property to determine whether the text frame contains text as shown in the following example.

```
Sub GetTextFromTextFrame()
    Dim shpText As Shape

    For Each shpText In ActiveDocument.Pages(1).Shapes
        If shpText.HasTextFrame = msoTrue Then
            With shpText.TextFrame
                If .HasText Then MsgBox .TextRange.Text
            End With
        End If
    Next
End Sub
```

Text frames can be linked together so that the text flows from the text frame of one shape into the text frame of another shape. Use the **NextLinkedTextFrame** and **PreviousLinkedTextFrame** properties to link text frames. The following

example creates a text box (a rectangle with a text frame) and adds some text to it. It then creates another text box and links the two text frames together so that the text flows from the first text frame into the second one.

```
Sub LinkTextBoxes()
    Dim shpTextBox1 As Shape
    Dim shpTextBox2 As Shape

    Set shpTextBox1 = ActiveDocument.Pages(1).Shapes.AddTextbox _
        (msoTextOrientationHorizontal, 72, 72, 72, 36)
    shpTextBox1.TextFrame.TextRange.Text = _
        "This is some text. This is some more text."

    Set shpTextBox2 = ActiveDocument.Pages(1).Shapes.AddTextbox _
        (msoTextOrientationHorizontal, 72, 144, 72, 36)
    shpTextBox1.TextFrame.NextLinkedTextFrame = shpTextBox2 _
        .TextFrame
End Sub
```



TextRange Object

Multiple objects └TextRange
└Multiple objects

Contains the text that's attached to a shape, as well as properties and methods for manipulating the text.

Using the TextRange Object

This topic describes how to:

- Return the text range in any shape you specify.
- Return a text range from the selection.
- Return particular characters, words, lines, sentences, or paragraphs from a text range.
- Insert text, the date and time, or the page number into a text range.

Return a text range from any shape you specify

Use the **TextRange** property of the **TextFrame** object to return a **TextRange** object for any shape you specify. Use the **Text** property to return the string of text in the **TextRange** object. The following example adds a rectangle to the active publication and sets the text it contains.

```
Sub AddTextToShape()
    With ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShapeRecta
        Left:=72, Top:=72, Width:=250, Height:=140)
        .TextFrame.TextRange.Text = "Here is some test text"
    End With
End Sub
```

Because the **Text** property is the default property of the **TextRange** object, the following two statements are equivalent.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange.text = "Here is some test text"
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange = "Here is some test text"
```

Use the **HasTextFrame** property to determine whether a shape has a text frame, and use the **HasText** property to determine whether the text frame contains text.

Return a text range from the selection

Use the **TextRange** property of the **Selection** object to return the currently selected text. The following example copies the selection to the Clipboard.

```
Sub CopyAndPasteText()
    With ActiveDocument
        .Selection.TextRange.Copy
        .Pages(1).Shapes(1).TextFrame.TextRange.Paste
    End With
End Sub
```

Return particular characters, words, lines, sentences, or paragraphs from a text range

Use one of the following methods to return a portion of the text of a **TextRange** object: **Characters**, **Lines**, **Paragraphs**, or **Words**. The following example formats the second word in the first shape on the first page of the active publication. For this example to work, the specified shape must contain text.

```
Sub FormatWords()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.Words(2).Font
        .Bold = msoTrue
        .Size = 15
        .Name = "Text Name"
    End With
End Sub
```

Inserting text, the date and time, or the page number into a text range

Use one of the following methods to insert characters into a **TextRange** object: **InsertAfter**, **InsertBefore**, **InsertDateTime**, **InsertPageNumber**, or **InsertSymbol**. This example inserts a new line with text after any existing text in the first shape on the first page of the active publication.

```
Sub InsertNewText()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange
        For intCount = 1 To 3
            .InsertAfter vbLf & "This is a test."
        Next intCount
    End With
End Sub
```

TextStyle Object

TextStyles └TextStyle
└Multiple objects

Represents a single built-in or user-defined style. The **TextStyle** object includes style attributes (font, font style, paragraph spacing, and so on) as properties of the **TextStyle** object. The **TextStyle** object is a member of the **TextStyles** collection. The **TextStyles** collection includes all the styles in the specified document.

Using the Style Object

Use **TextStyles**(*index*), where *index* is the text style number or name, to return a single **TextStyle** object. You must exactly match the spelling and spacing of the style name, but not necessarily its capitalization.

The following example displays the style name and base style of the first style in the **TextStyles** collection.

```
Sub BaseStyleName()
    With ActiveDocument.TextStyles(1)
        MsgBox "Style name= " & .Name _
            & vbCr & "Base style= " & .BaseStyle
    End With
End Sub
```

Use the **Add** method to create a new style. To apply a style to a range, paragraph, or multiple paragraphs, set the **TextStyle** property to a user-defined or built-in style name. The following example creates a new style and applies it to the paragraph at the insertion point position.

```
Sub ApplyTextStyle()
    Dim styNew As TextStyle
    Dim fntStyle As Font

    'Create a new style
    Set styNew = ActiveDocument.TextStyles.Add(StyleName:="NewStyle"
    Set fntStyle = styNew.Font

    'Format the Font object
    With fntStyle
        .Name = "Tahoma"
        .Size = 20
        .Bold = msoTrue
    End With

    'Apply the Font object formatting to the new style
    styNew.Font = fntStyle

    'Apply the new style to the selected paragraph
    Selection.TextRange.ParagraphFormat.TextStyle = "NewStyle"
End Sub
```

ThreeDFormat Object

Multiple objects └ThreeDFormat
└ColorFormat

Represents a shape's three-dimensional formatting.

Using The ThreeDFormat Object

Use the **ThreeD** property to return a **ThreeDFormat** object. This example sets the depth, extrusion color, extrusion direction, and lighting direction for the 3-D effects applied to shape one in the active publication.

```
Sub SetThreeDSettings()
    Dim tdfTemp As ThreeDFormat

    Set tdfTemp = _
        ActiveDocument.Pages(1).Shapes(1).ThreeD

    With tdfTemp
        .Visible = True
        .Depth = 50
        .ExtrusionColor.RGB = RGB(255, 100, 255)
        .SetExtrusionDirection _
            PresetExtrusionDirection:=msoExtrusionTop
        .PresetLightingDirection = msoLightingLeft
    End With
End Sub
```

Remarks

You cannot apply three-dimensional formatting to some kinds of shapes, such as beveled shapes. Most of the properties and methods of the **ThreeDFormat** object for such a shape will fail.

View Object

Document └View
└Page

Contains the view attributes (show all, field shading, table gridlines, and so on) for a window or pane.

Using the View Object

Use the **ActiveView** property to return the **View** object. The following example specifies the zoom setting.

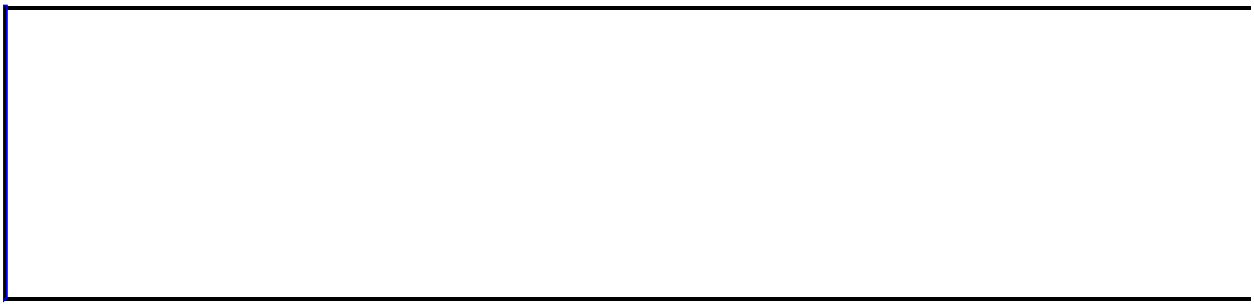
```
Sub ZoomFitSelection()
    ActiveDocument.ActiveView.Zoom = pbZoomFitSelection
End Sub
```

The following examples zoom in and out, respectively, on the active view.

```
Sub ViewZoomIn()
    ActiveDocument.ActiveView.ZoomIn
End Sub

Sub ViewZoomOut()
    ActiveDocument.ActiveView.ZoomOut
End Sub
```

The following example scrolls the active view to the specified shape.

```
Sub ScrollToShape()
    Dim shpOne As Shape

    Set shpOne = ActiveDocument.Pages(1).Shapes(1)
    ActiveDocument.ActiveView.ScrollShapeIntoView Shape:=shpOne
End Sub
```



WebCheckBox Object

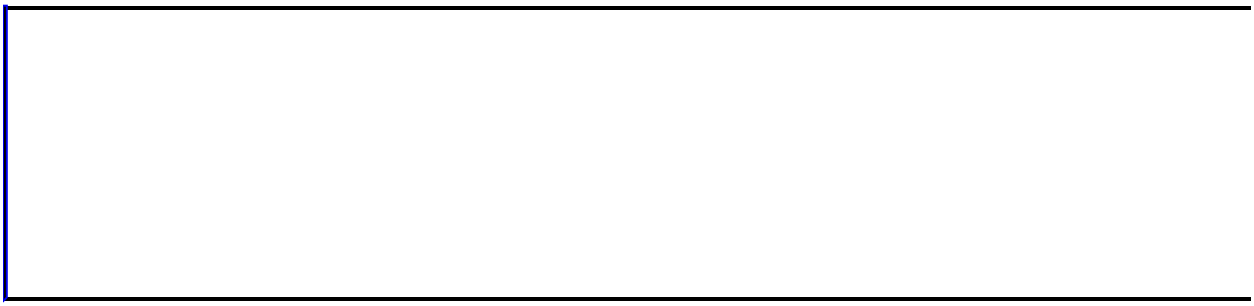
Shape
└WebCheckBox

Represents a Web check box control. The **WebCheckBox** object is a member of the **Shape** object.

Using the WebCheckBox object

Use the **AddWebControl** method to create new Web check box. Use the **WebCheckBox** property to access a Web check box control shape. This example creates a new Web check box and specifies that its default state is checked; then it adds a text box next to it to describe it.

```
Sub CreateNewWebCheckBox()
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl(Type:=pbWebControlCheckBox, Left:=100, _
                Top:=123, Width:=17, Height:=12).WebCheckBox
            .Selected = msoTrue
        End With
        With .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
                Left:=118, Top:=120, Width:=70, Height:=15)
            .TextFrame.TextRange.Text = "Power User?"
        End With
    End With
End Sub
```



WebCommandButton Object

Shape └WebCommandButton
└WebHiddenFields

Represents a Web command button control. The **WebCommandButton** object is a member of the **Shape** object.

Using the WebCommandButton object

Use the **AddWebControl** method to create new Web command button. Use the **WebCommandButton** property to access a Web command button control shape. This example creates a Web form Submit command button and sets the script path and file name to run when a user clicks the button.

```
Sub CreateActionWebButton()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlCommandButton, Left:=150, _
            Top:=150, Width:=75, Height:=36).WebCommandButton
        .ButtonText = "Submit"
        .ButtonType = pbCommandButtonSubmit
        .ActionURL = "http://www.tailspintoys.com/" _
            & "scripts/ispscript.cgi"
    End With
End Sub
```

WebListBox Object

Shape
└WebListBox
└WebListBoxItems

Represents a Web list box control. The **WebListBox** object is a member of the **Shape** object.

Using the WebListBox object

Use the **AddWebControl** method to create a new Web list box. Use the **WebListBox** property to access a Web list box control shape. Use the **AddItem** method of the **WebListBoxItems** object to add items to a Web list box. This example creates a new Web list box and adds several items to it. Note that when initially created, a Web list box control contains three default items. This example includes a routine that deletes the default list box items before adding new items.

Note When you create a Web list box, its initial width is 300 points. However, Microsoft Publisher automatically changes this width based on the width of the items in the list.

```
Sub CreateWebListBox()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl(Type:=pbWebControlListBox, Left:=100, _
                Top:=150, Width:=300, Height:=72).WebListBox
            .MultiSelect = msoFalse
            With .ListBoxItems
                For intCount = 1 To .Count
                    .Delete (1)
                Next
                .AddItem Item:="Green"
                .AddItem Item:="Purple"
                .AddItem Item:="Red"
                .AddItem Item:="Black"
            End With
        End With
    End With
End Sub
```

WebNavigationBarSet Object

WebNavigationBarSets └WebNavigationBarSet
└WebNavigationBarHyperlinks

Represents a Web navigation bar set for the current document. The **WebNavigationBarSet** object is a member of the **WebNavigationBarSets** collection, which includes all of the Web navigation bar sets in the current document.

Using the WebNavigationBarSet Object

Use **WebNavigationBarSet**.**AddToEveryPage**(*Left*, *Top*, [*Width*]), where *Left* is the position of the left edge of the shape, *Top* is the position of the top edge of the shape, and *Width* is the width of the shape representing the Web navigation bar set, to add the specified Web navigation bar to every page of a document. The following example adds the first Web navigation bar set to every page that has the **AddHyperlinkToWebNavbar** property set to **True** when adding the page or the **Page.WebPageOptions.IncludePageOnNewWebNavigationBars** property set to **True**.

```
Dim objWebNavBarSet as WebNavigationBarSet
Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets(1)
objWebNavBarSet.AddToEveryPage Left:=50, Top:=10, Width:=500
```

Use **WebNavigationBarSet**.**DeleteSetAndInstances** to remove the Web navigation bar set and every instance of it from the document. The following example deletes all instances of each **WebNavigationBarSet** object in the **WebNavigationBarSets** collection.

```
Dim objWebNavBarSet As WebNavigationBarSet
For Each objWebNavBarSet In ActiveDocument.WebNavigationBarSets
    objWebNavBarSet.DeleteSetAndInstances
Next objWebNavBarSet
```

There are three properties that concern horizontally oriented Web navigation bars. Use **WebNavigationBarSet**.**IsHorizontal** to determine the orientation of the navigation bar set. The **ChangeOrientation** method is used to set the orientation of the Web navigation bar set. If the orientation is set to **horizontal**, **HorizontalAlignment** and **HorizontalButtonCount** properties can then be set. The following example adds the first navigation bar in the **WebNavigationBarSets** collection of the active document to each page that has the **AddHyperlinkToWebNavbar** property set to **True** or the **Page.WebPageOptions.IncludePageOnNewWebNavigationBars** property set to **True**, and then sets the button style to **small**. A test is performed to determine whether the navigation bar set is horizontal or not. If it is not, the **ChangeOrientation** method is called and the orientation is set to **horizontal**. After the navigation bar is oriented horizontally, the horizontal button count is set to **3** and the horizontal alignment of the buttons is set to **left**.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets(1)
With objWebNav
    .AddToEveryPage Left:=10, Top:=10
    If .IsHorizontal = False Then
        .ChangeOrientation pbNavBarOrientHorizontal
    End If
    .HorizontalButtonCount = 3
    .HorizontalAlignment = pbnbAlignLeft
End With
```

WebOptionButton Object

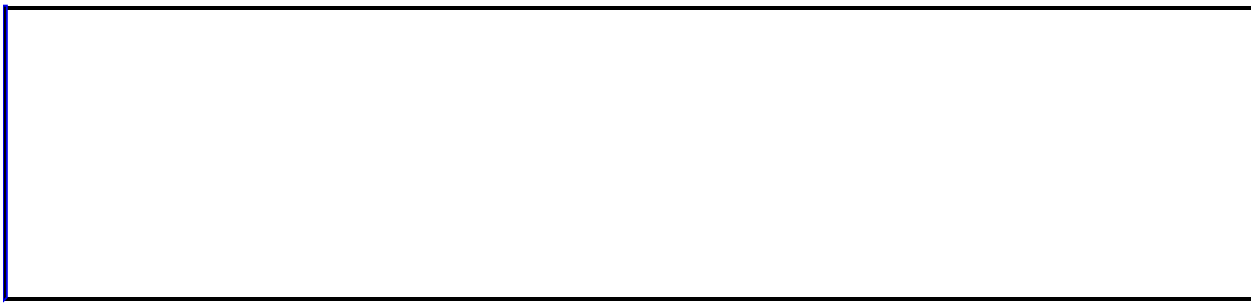
Shape └WebOptionButton

Represents a Web option button control. The **WebOptionButton** object is a member of the **Shape** object.

Using the WebOptionButton object

Use the **AddWebControl** method to create new Web option button. Use the **WebOptionButton** property to access a Web option button control shape. This example creates a new Web option button and specifies that its default state is selected; then it adds a text box next to it to describe it.

```
Sub CreateNewWebOptionButton()
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl(Type:=pbWebControlOptionButton, Left:=10
                Top:=123, Width:=16, Height:=10).WebOptionButton
            .Selected = msoTrue
        End With
        With .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
                Left:=120, Top:=120, Width:=70, Height:=15)
            .TextFrame.TextRange.Text = "Advanced User"
        End With
    End With
End Sub
```



WebOptions Object

Application └WebOptions

Represents the properties of a Web publication, including options for saving and encoding the publication, and enabling Web-safe fonts and font schemes. The **WebOptions** object is a member of the **Application** object.

Using the WebOptions Object

Use the **WebOptions** property on the **Application** object to return a **WebOptions** object. The following example sets an object variable equal to Publisher's **WebOptions** object.

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions
```

The properties of the **WebOptions** object are used to specify the behavior of Web publications. This means that when any of these properties are modified, newly created Web publications will inherit the modified properties.

Note that the **WebOptions** object is available from print publications as well as Web publications. However, the properties of this object have no effect on print publications.

WebPageOptions Object

Page └WebPageOptions

Represents the properties of a single Web page within a Web publication, including options for adding the title and description of the page, background sounds, in addition to other options. The **WebPageOptions** object is a member of the **Page** object.

Using the WebPageOptions Object

Use the **WebPageOptions** property on the **Page** object to return a **WebPageOptions** object. Use the **Description** property to set the description of a specified Web page. The following example sets the description for the second page of the active Web publication.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(2).WebPageOptions

With theWPO
    .Description = "Company Profile"
End With
```

Note that the **WebPageOptions** object is only available when the active publication is a Web publication. A run-time error is returned if trying to access this object from a print publication.

WebTextBox Object

Shape └WebTextBox

Represents a Web text box control. The **WebTextBox** object is a member of the **Shape** object.

Using the WebTextBox object

Use the **AddWebControl** method to create new Web option button. Use the **WebTextBox** property to access a Web text box control shape. This example creates a new Web text box, specifies default text, indicates that entry is required, and limits entry to 50 characters.

```
Sub CreateWebTextBox()
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl(Type:=pbWebControlSingleLineTextBox, _
                Left:=100, Top:=100, Width:=150, Height:=15).WebText
            .DefaultText = "Please Enter Your Full Name"
            .RequiredControl = msoTrue
            .Limit = 50
        End With
    End With
End Sub
```

Window Object

Multiple objects └Window

Represents a window. Many publication characteristics, such as scroll bars and rulers, are actually properties of the window.

Using the Window Object

Use the **ActiveWindow** property to return a **Window** object. The following example maximizes the active window.

```
Sub MaximizeWindow
    ActiveWindow.WindowState = pbWindowStateMaximize
End Sub
```

Use the **Caption** property to return the file and application names of the active window. The following example displays a message with the file name and Microsoft Publisher application name.

```
Sub ShowFileApNames
    MsgBox Windows(1).Caption
End Sub
```

Wizard Object

Multiple objects └Wizard
└WizardProperties

Represents the publication design associated with a publication or the wizard associated with a Design Gallery object.

Using the Wizard object

Use the **Wizard** property of a **Document**, **Page**, **Shape** or **ShapeRange** object to return a **Wizard** object. The following example reports on the publication design associated with the active publication, displaying its name and current settings.

```
Dim wizTemp As Wizard
Dim wizproTemp As WizardProperty
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp
    Set wizproAll = .Properties
    MsgBox "Publication Design associated with " _
        & "current publication: " _
        & .Name
    For Each wizproTemp In wizproAll
        With wizproTemp
            MsgBox "   Wizard property: " _
                & .Name & " = " & .CurrentValueId
        End With
    Next wizproTemp
End With
```

Note Depending on the language version of Publisher that you are using, you may receive an error when using the above code. If this occurs, you will need to build in error handlers to circumvent the errors. The following example functions as the code above but has error handlers built in for this situation.

```
Sub ExampleWithErrorHandlers()
    Dim wizTemp As Wizard
    Dim wizproTemp As WizardProperty
    Dim wizproAll As WizardProperties

    Set wizTemp = ActiveDocument.Wizard

    With wizTemp
        Set wizproAll = .Properties
        Debug.Print "Publication Design associated with " _
            & "current publication: " _
            & .Name
```

```
        For Each wizproTemp In wizproAll
            With wizproTemp
                If wizproTemp.Name = "Layout" Or wizproTemp _
                        .Name = "Layout (Intl)" Then
                    On Error GoTo Handler
                    MsgBox "   Wizard property: " _
                        & .Name & " = " & .CurrentValueId

Handler:
                    If Err.Number = 70 Then Resume Next
                Else
                    MsgBox "   Wizard property: " _
                        & .Name & " = " & .CurrentValueId
                End If
            End With
        Next wizproTemp
    End With
End Sub
```

WizardProperty Object

Wizard └WizardProperties
└WizardProperty
└WizardValues

Represents a setting that is part of a specific publication design or a Design Gallery object's wizard.

Using the WizardProperty object

Use the **Item** property or the **FindByPropertyID** method with the **WizardProperties** collection to return a single **WizardProperty** object. The following example reports on the publication design associated with the active publication, displaying its name and current settings.

```
Dim wizTemp As Wizard
Dim wizproTemp As WizardProperty
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp
    Set wizproAll = .Properties
    Debug.Print "Publication Design associated with " _
        & "current publication: " _
        & .Name
    For Each wizproTemp In wizproAll
        With wizproTemp
            Debug.Print "   Wizard property: " _
                & .Name & " = " & .CurrentValueId
        End With
    Next wizproTemp
End With
```

Note Depending on the language version of Publisher that you are using, you may receive an error when using the above code. If this occurs, you will need to build in error handlers to circumvent the errors. For more information, see **Wizard Object**.

WizardValue Object

WizardProperty └WizardValues
└WizardValue

Represents a possible value for the specified wizard property.

Using the WizardValue object

Use the **Item** property of the **WizardValues** collection to return a **WizardValue** object. The following example displays the current value for the first wizard property in the active publication and then lists all the other possible values.

```
Dim valAll As WizardValues
Dim valLoop As WizardValue

With ActiveDocument.Wizard
    Set valAll = .Properties(1).Values

    MsgBox "Wizard: " & .Name & vbLf & _
        "Property: " & .Properties(1).Name & vbLf & _
        "Current value: " & .Properties(1).CurrentValueId

    For Each valLoop In valAll
        MsgBox "Possible value: " & valLoop.ID & " (" & valLoop.Name
    Next valLoop
End With
```

WrapFormat Object

Multiple objects └WrapFormat

Represents all the properties for wrapping text around a shape or shape range.

Using the WrapFormat Object

Use the **TextWrap** property to return a **WrapFormat** object. The following example adds an oval to the active publication and specifies that publication text wrap around the left and right sides of the square that circumscribes the oval. There will be a 0.1-inch margin between the publication text and the top, bottom, left side, and right side of the square.

```
Sub SetTextWrapFormatProperties()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoS
         Left:=36, Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```

Activate Method

Activates a window or OLE object.

expression.**Activate**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Because Publisher runs in a single window, using the **Activate** method with a **Window** object makes Publisher the active application.

Example

The following example makes Publisher the active application.

```
Application.ActiveWindow.Activate
```

The following example adds an Excel spreadsheet to the first page of the active publication and activates the spreadsheet for editing.

```
Dim shpSheet As Shape

Set shpSheet = ActiveDocument.Pages(1).Shapes.AddOLEObject _
    (Left:=72, Top:=72, ClassName:="Excel.Sheet")

shpSheet.OLEFormat.Activate
```


Show All

Add Method

 **Add** method as it applies to the **Columns** object.

Adds a new **Column** object to the specified **Columns** object and returns the new **Column** object.

expression.**Add**(***BeforeColumn***)

expression Required. An expression that returns a **Columns** object.

BeforeColumn Optional **Long**. The number of the column before which to insert the new column. If this argument is omitted, the new column is added after the existing columns. An error occurs if the value of this argument does not correspond to an existing column in the table.

 **Add** method as it applies to the **Hyperlinks** object.

Adds a new **Hyperlink** object to the specified **Hyperlinks** object and returns the new **Hyperlink** object.

expression.**Add**(***Text***, ***Address***, ***RelativePage***, ***PageID***, ***TextToDisplay***)

expression Required. An expression that returns a **Hyperlinks** object.

Text Required **TextRange** object. The text range to be converted into a hyperlink.

Address Optional **String**. The address of the new hyperlink. If ***RelativePage*** is **pbHlinkTargetTypeURL** (default) or **pbHlinkTargetTypeEmail**, ***Address*** must be specified or an error occurs.

RelativePage Optional **PbHlinkTargetType**. The type of hyperlink to add.

PbHlinkTargetType can be one of these PbHlinkTargetType constants.
pbHlinkTargetTypeEmail
pbHlinkTargetTypeFirstPage

pbHlinkTargetTypeLastPage
pbHlinkTargetTypeNextPage
pbHlinkTargetTypeNone Not supported.
pbHlinkTargetTypePageID
pbHlinkTargetTypePreviousPage
pbHlinkTargetTypeURL *default*

PageID Optional **Long**. The page ID of the destination page for the new hyperlink. If ***RelativePage*** is **pbHlinkTargetTypePageID**, ***PageID*** must be specified or an error occurs. The page ID corresponds to the **PageID** property of the destination page.

TextToDisplay Optional **String**. The display text of the new hyperlink. If specified, ***TextToDisplay*** replaces the text range specified by the ***Text*** argument.

 **Add** method as it applies to the **MailMergeFilters** object.

Adds a new filter criterion to the specified **MailMergeFilters** object.

expression.**Add**(***Column***, ***Comparison***, ***Conjunction***, ***bstrCompareTo***, ***DeferUpdate***)

expression Required. An expression that returns a **MailMergeFilters** object.

Column Required **String**. The name of the table in the data source.

Comparison Required **MsoFilterComparison**. How the data in the table is filtered.

MsoFilterComparison can be one of these MsoFilterComparison constants.
msoFilterComparisonContains
msoFilterComparisonEqual
msoFilterComparisonGreaterThan
msoFilterComparisonGreaterThanEqual
msoFilterComparisonIsBlank
msoFilterComparisonIsNotBlank
msoFilterComparisonLessThan

msoFilterComparisonLessThanEqual
msoFilterComparisonNotContains
msoFilterComparisonNotEqual

Conjunction Required **MsoFilterConjunction**. Determines how this filter relates to other filters in the **MailMergeFilters** object.

MsoFilterConjunction can be one of these MsoFilterConjunction constants.
msoFilterConjunctionAnd
msoFilterConjunctionOr

bstrCompareTo Optional **String**. If the ***Comparison*** argument is something other than **msoFilterComparisonIsBlank** or **msoFilterComparisonIsNotBlank**, a string to which the data in the table is compared.

DeferUpdate Optional **Boolean**. **True** to queue the filters and apply them when the **ApplyFilter** method is called. **False** to apply the filter condition immediately. Default is **False**.

 **Add** method as it applies to the **MasterPages** object.

Adds a new **Page** object to the specified **MasterPages** object and returns the new **Page** object.

expression.**Add**([***IsTwoPageMaster***], [***Abbreviation***], [***Description***],)

expression Required. An expression that returns a **MasterPages** object.

IsTwoPageMaster Optional **Boolean**. **True** if the master page will be part of a two page spread.

Abbreviation Optional **String**. The abbreviation, or short name, for the master page. An error occurs if this is not unique.

Description Optional **String**. The description for the master page.

 **Add** method as it applies to the **Pages** object.

Adds a new **Page** object to the specified **Pages** object and returns the new **Page** object.

expression.**Add**(***Count***, ***After***, [***DuplicateObjectsOnPage***], [***AddHyperlinkToWebNavBar***])

expression Required. An expression that returns a **Pages** object.

Count Required **Long**. The number of new pages to add.

After Required **Long**. The page index of the page after which to add the new pages. A zero for this argument adds new pages at the beginning of the publication.

DuplicateObjectsOnPage Optional **Long**. The page index of the page from which objects should be copied to the new pages. If this argument is omitted, the new pages will be blank.

AddHyperlinkToWebNavBar Optional **Boolean**. Specifies whether links to the new pages will be added to the automatic navigation bars of existing pages. If **True**, links to the new pages will be added to the automatic navigation bars of existing pages only. If **False**, links to the new pages will not be added to the automatic navigation bars of existing pages or new pages added in the future. Default is **False**.

 **Add** method as it applies to the **Plates** object.

Adds a new color plate to the specified **Plates** object.

expression.**Add**(***PlateColor***)

expression Required. An expression that returns a **Plates** object.

PlateColor Optional **ColorFormat** object. The color settings to apply to the new plate.

Remarks

If the **ColorMode** property of the specified publication is not **pbColorModeSpot** or **pbColorModeSpotAndProcess**, an error occurs.

 **Add** method as it applies to the **Rows** object.

Adds a new **Row** object to the specified **Rows** object and returns the new **Row** object.

expression.**Add**(***BeforeRow***)

expression Required. An expression that returns a **Rows** object.

BeforeRow Optional **Long**. The number of the row before which to insert the new row. If this argument is omitted, the new row is added after the existing rows. An error occurs if the value of this argument does not correspond to an existing row in the table.

 **Add** method as it applies to the **RulerGuides** object.

Adds a new ruler guide to the specified **RulerGuides** object.

expression.**Add**(***Position***, ***Type***)

expression Required. An expression that returns a **RulerGuides** object.

Position Required **Variant**. The position relative to the left edge or top edge of the page where the new ruler guide will be added. Numeric values are evaluated in points; strings are evaluated in the units specified and can be in any measurement unit supported by Microsoft Publisher (for example, "2.5 in").

Type Required **PbRulerGuideType**. The type of ruler guide to add.

PbRulerGuideType can be one of these PbRulerGuideType constants.
pbRulerGuideTypeHorizontal
pbRulerGuideTypeVertical

 **Add** method as it applies to the **TabStops** object.

Adds a new tab stop to the specified **TabStops** object.

expression.**Add**(***Position***, ***Alignment***, ***Leader***)

expression Required. An expression that returns a **TabStops** object.

Position Required **Variant**. The horizontal position of the new tab stop relative to the left edge of the text frame. Numeric values are evaluated in points; strings are evaluated in the units specified and can be in any measurement unit supported by Microsoft Publisher (for example, "2.5 in").

Alignment Required **PbTabAlignmentType**. The alignment setting for the tab stop.

PbTabAlignmentType can be one of these PbTabAlignmentType constants.
pbTabAlignmentCenter
pbTabAlignmentDecimal
pbTabAlignmentLeading
pbTabAlignmentTrailing

Leader Required **PbTabLeaderType**. The type of leader for the tab stop.

PbTabLeaderType can be one of these PbTabLeaderType constants.
pbTabLeaderBullet
pbTabLeaderDashes
pbTabLeaderDot
pbTabLeaderLine
pbTabLeaderNone

 **Add** method as it applies to the **Tags** object.

Adds a new **Tag** object to the specified **Tags** object and returns the new **Tag** object.

expression.**Add**(***Name***, ***Value***)

expression Required. An expression that returns a **Tags** object.

Name Required **String**. The name of the tag to add. If a tag already exists with the same name, an error occurs.

Value Required **Variant**. The value to assign to the tag.

 **Add** method as it applies to the **TextStyles** object.

Adds a new **TextStyle** object to the specified **TextStyles** object and returns the new **TextStyle** object.

expression.**Add**(***Font***, ***ParagraphFormat***, ***StyleName***, ***BasedOn***)

expression Required. An expression that returns a **TextStyles** object.

StyleName Required **String**. The name of the new text style. If the name matches an existing text style, the existing text style is overwritten.

BasedOn Optional **String**. The name of the text style on which the new text style is based. If the name does not match an existing text style, an error occurs.

Font Optional **Font** object. The font settings to apply to the new text style.

ParagraphFormat Optional **ParagraphFormat** object. The paragraph formatting to apply to the new text style.

 **Add** method as it applies to the **WebHiddenFields** object.

Adds a new hidden field to a Web form and returns a **Long** indicating the number of the new field in the **WebHiddenFields** collection. New fields are always placed at the end of the current field list.

expression.**Add**(***Name***, ***Value***)

expression Required. An expression that returns a **WebHiddenFields** object.

Name Required **String**. The name of the new field.

Value Required **String**. The value of the new field.

Example

 As it applies to the **Columns** object.

The following example adds a column before column three in the specified table.

```
Dim colNew As Column

Set colNew = ActiveDocument.Pages(1).Shapes(1) _
    .Table.Columns.Add(BeforeColumn:=3)
```

 As it applies to the **Hyperlinks** object.

The following example adds hyperlinks to shapes one and two on page one of the active publication. The first hyperlink points to an external website, and the second link points to the fourth page in the publication. Shapes one and two must be text boxes and there must be at least four pages in the publication for this example to work.

```
Dim hypNew As Hyperlink
Dim lngPageID As Long
Dim strPage As String

With ActiveDocument.Pages(1).Shapes(1).TextFrame
    Set hypNew = .TextRange.Hyperlinks.Add(Text:=.TextRange, _
        Address:="http://www.tailspintoys.com/", _
        TextToDisplay:="Tailspin")
End With

lngPageID = ActiveDocument.Pages(4).PageID
strPage = "Go to page " _
    & Str(ActiveDocument.Pages(4).PageNumber)

With ActiveDocument.Pages(1).Shapes(2).TextFrame
    Set hypNew = .TextRange.Hyperlinks.Add(Text:=.TextRange, _
        RelativePage:=pbHlinkTargetTypePageID, _
        PageID:=lngPageID, _
        TextToDisplay:=strPage)
End With
```

 As it applies to the **MasterPages** object.

The following example adds a new master page to the active document.

```
ActiveDocument.MasterPages.Add _
    IsTwoPageMaster:=False, _
    Abbreviation:="X", _
    Description:="Master Page X"
```

 As it applies to the **Pages** object.

The following example adds four new pages after the first page in the publication and copies all the objects from the first page to the new pages.

```
Dim pgNew As Page

Set pgNew = ActiveDocument.Pages _
    .Add(Count:=4, After:=1, DuplicateObjectsOnPage:=1)
```

The following example demonstrates adding two new pages to the publication and setting the ***AddHyperlinkToWebNavBar*** parameter to **True** for these two pages. This specifies that links to these two new pages be added to the automatic navigation bars of existing pages and those added in the future.

Another page is then added to the publication, and the ***AddHyperlinkToWebNavBar*** is omitted. This means that the **IncludePageOnNewWebNavigationBars** property is **False** for the newly added page, and links to this page will not be included in the automatic navigation bars of existing pages.

```
Dim thePage As page
Dim thePage2 As page

Set thePage = ActiveDocument.Pages.Add(Count:=2, _
    After:=4, AddHyperlinkToWebNavBar:=True)

Set thePage2 = ActiveDocument.Pages.Add(Count:=1, After:=6)
```

 As it applies to the **Plates** object.

The following example adds a color plate to the active publication if it is a spot-color publication.

```
If ActiveDocument.ColorMode = pbColorModeSpot Then
    ActiveDocument.Plates.Add
End If
```

 As it applies to the **Rows** object.

The following example adds a row before row three in the specified table.

```
Dim rowNew As Row

Set rowNew = ActiveDocument.Pages(1).Shapes(1) _
    .Table.Rows.Add(BeforeRow:=3)
```

 As it applies to the **RulerGuides** object.

The following example adds ruler guides to page one that are 0.5 inches from the left and top edges of the page.

```
With ActiveDocument.Pages(1).RulerGuides
    .Add Position:="0.5 in", Type:=pbRulerGuideTypeHorizontal
    .Add Position:="0.5 in", Type:=pbRulerGuideTypeVertical
End With
```

 As it applies to the **TabStops** object.

The following example adds a new left-aligned tab stop 0.5 inches from the left edge of the specified text frame.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange.ParagraphFormat.Tabs _
    .Add Position:="0.5 in", _
    Alignment:=pbTabAlignmentLeading, _
    Leader:=pbTabLeaderNone
```

 As it applies to the **Tags** object.

The following example adds a tag to shape one on page one of the active publication.

```
Dim tagNew As Tag

Set tagNew = ActiveDocument.Pages(1).Shapes(1).Tags _
    .Add(Name:="required", Value:="yes")
```

 As it applies to the **TextStyles** object.

The following example adds a new text style to the active publication based on the Normal text style.

```
Dim tsNew As TextStyle

Set tsNew = ActiveDocument.TextStyles _
    .Add(StyleName:="Title", BasedOn:="Normal")
```

 As it applies to the **WebHiddenFields** object.

The following example adds a new hidden field to the specified Web command button control. Shape one on page one of the active publication must be a Web command button control for this example to work.

```
ActiveDocument.Pages(1).Shapes(1) _
    .WebCommandButton.HiddenFields _
    .Add Name:="subject", Value:="service request"
```


Show All

AddCallout Method

Adds a new **Shape** object representing a borderless line callout to the specified **Shapes** collection.

expression.**AddCallout**(***Type***, ***Left***, ***Top***, ***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

Type Required **MsoCalloutType**. The type of callout line.

MsoCalloutType can be one of these MsoCalloutType constants.
msoCalloutOne A horizontal or vertical single-segment callout line.
msoCalloutTwo A freely-rotating single-segment callout line.
msoCalloutThree A two-segment callout line.
msoCalloutFour A three-segment callout line.
msoCalloutMixed Not used for this method.

Left Required **Variant**. The position of the left edge of the shape representing the line callout.

Top Required **Variant**. The position of the top edge of the shape representing the line callout.

Width Required **Variant**. The width of the shape representing the line callout.

Height Required **Variant**. The height of the shape representing the line callout.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a new freely-rotating callout line to the first page of the active publication.

```
Dim shpCallout As Shape

Set shpCallout = ActiveDocument.Pages(1).Shapes.AddCallout _
    (Type:=msoCalloutTwo, _
    Left:=144, Top:=216, _
    Width:=36, Height:=72)
```


Show All

AddCatalogMergeArea Method

Adds a **Shape** object that represents the specified publication's catalog merge area.

expression.**AddCatalogMergeArea**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Only one catalog merge area can be added to a publication page. Typically, a publication will only have one catalog merge area.

Although you can add one catalog merge area per publication page, you can only connect to a single data source for a publication. What data is merged is determined by the catalog merge area on the active page, and the data fields it contains.

Use the **AddToCatalogMergeArea** method of the **Shape** or **ShapeRange** objects to add shapes to a catalog merge area.

Use the **Insert** method of the **MailMergeDataFields** collection to add a picture data field to a publication's catalog merge area.

Use the **InsertMailMergeField** method of the **TextRange** object to add a text data field to a text box in the publication's catalog merge area.

Use the **RemoveCatalogMergeArea** method of the **Shape** object to remove a catalog merge area from a publication.

This method corresponds to selecting a catalog merge in **Step 1: Select a merge type** of the **Mail and Catalog Merge Wizard**.

Example

The following example adds a catalog merge area to the first page of the specified publication.

```
ThisDocument.Pages(1).Shapes.AddCatalogMergeArea
```


Show All

AddConnector Method

Adds a new **Shape** object representing a connector to the specified **Shapes** collection.

expression.**AddConnector**(***Type***, ***BeginX***, ***BeginY***, ***EndX***, ***EndY***)

expression Required. An expression that returns one of the objects in the Applies To list.

Type Required **MsoConnectorType**. The type of connector to add.

MsoConnectorType can be one of these MsoConnectorType constants.
msoConnectorCurve Adds a curved connector.
msoConnectorElbow Adds an elbow-shaped connector.
msoConnectorStraight Adds a straight-line connector.
msoConnectorTypeMixed Not used with this method.

BeginX Required **Variant**. The x-coordinate of the beginning point of the connector.

BeginY Required **Variant**. The y-coordinate of the beginning point of the connector.

EndX Required **Variant**. The x-coordinate of the ending point of the connector.

EndY Required **Variant**. The y-coordinate of the ending point of the connector.

Remarks

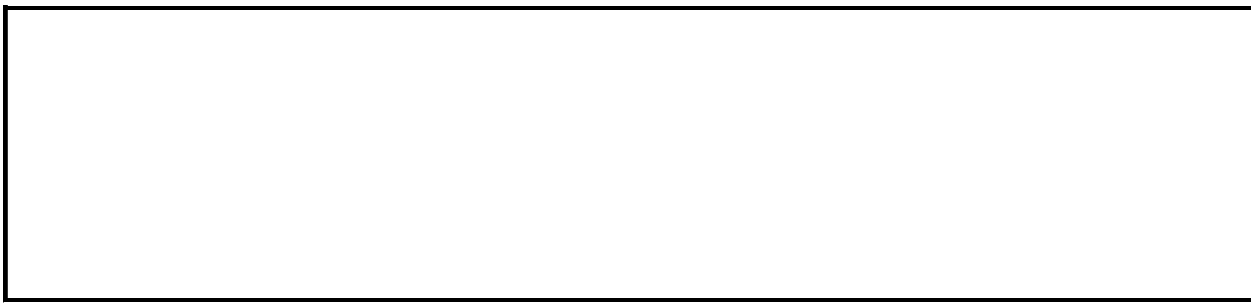
For the ***BeginX***, ***BeginY***, ***EndX***, and ***EndY*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

The new connector isn't connected to any other shape; use the **BeginConnect** and **EndConnect** methods to connect the new connector to another shape.

Example

The following example adds a new straight-line connector to the first page of the active publication.

```
Dim shpConnect As Shape

Set shpConnect = ActiveDocument.Pages(1).Shapes.AddC onnector _
    (Type:=msoConnectorStraight, _
    BeginX:=144, BeginY:=144, _
    EndX:=180, EndY:=72)
```



AddCurve Method

Adds a new **Shape** object representing a Bézier curve to the specified **Shapes** collection.

expression.**AddCurve**(***SafeArrayOfPoints***)

expression Required. An expression that returns one of the objects in the Applies To list.

SafeArrayOfPoints Required **Variant**. An array of coordinate pairs that specifies the vertices and control points of the curve. The first point you specify is the starting vertex, and the next two points are control points for the first Bézier segment. Then, for each additional segment of the curve, you specify a vertex and two control points. The last point you specify is the ending vertex for the curve. Note that you must always specify 3*n* + 1 points, where *n* is the number of segments in the curve.

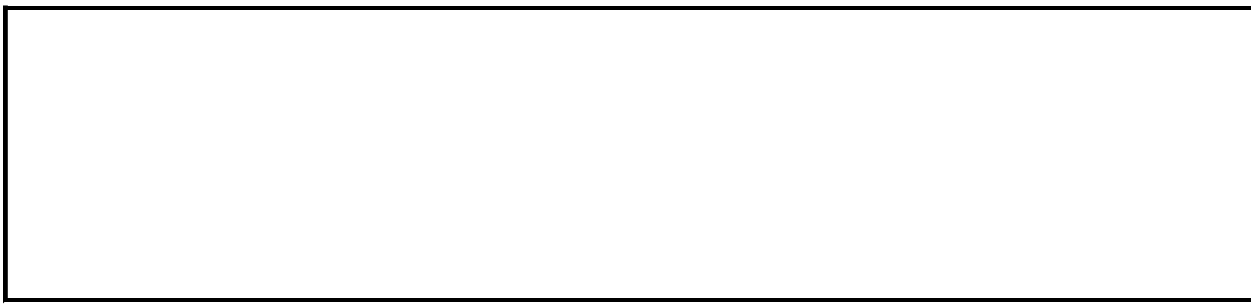
Remarks

For the array elements in ***SafeArrayOfPoints***, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a two-segment Bézier curve to the first page of the active publication.

```
Dim shpCurve As Shape
Dim arrPoints(1 To 4, 1 To 2) As Single

arrPoints(1, 1) = 0
arrPoints(1, 2) = 0
arrPoints(2, 1) = 72
arrPoints(2, 2) = 72
arrPoints(3, 1) = 144
arrPoints(3, 2) = 36
arrPoints(4, 1) = 216
arrPoints(4, 2) = 108

Set shpCurve = ActiveDocument.Pages(1).Shapes.AddCurve _
    (SafeArrayOfPoints:=arrPoints)
```



AddEmptyPictureFrame Method

Returns a **Shape** object that represents an empty picture frame inserted at the specified coordinates.

expression.**AddEmptyPictureFrame**(***Left***, ***Top***, [***Width* = *-1***], [***Height* = *-1***])

expression Required. An expression that returns a **Shapes** collection.

Left Required **Variant**. The position of the left edge of the shape representing the picture.

Top Required **Variant**. The position of the top edge of the shape representing the picture.

Width Required **Variant**. The width of the shape representing the picture. Default is -1, meaning that the width of the shape is automatically set to 54 points if the parameter is left blank.

Height Required **Variant**. The height of the shape representing the picture. Default is -1, meaning that the height of the shape is automatically set to 54 points if the parameter is left blank.

Remarks

For **Left**, **Top**, **Width**, and **Height** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "1.5 in").

The blank picture frame has the default ToolTip "Empty Picture Frame". This is changed to "Picture" when an image is selected for the **Shape**.

Example

This example places an empty picture frame in the center of the first page of the publication and rotates it by 45 degrees. The **AlternativeText** property is set to "Picture Placeholder 1" for the Web.

```
Dim shpPlaceholder As Shape

Set shpPlaceholder = _
    ActiveDocument.Pages(1).Shapes.AddEmptyPictureFrame( _
    230, 320, 150, 150)

With shpPlaceholder
    .AlternativeText = "Picture Placeholder 1"
    .Rotation = 45
End With
```


Show All

AddGroupWizard Method

Adds a **Shape** object representing a Design Gallery object to the publication.

expression.**AddGroupWizard**(***Wizard***, ***Left***, ***Top***, ***Width***, ***Height***, ***Design***)

expression Required. An expression that returns one of the objects in the Applies To list.

Wizard Required **PbWizardGroup**. The type of Design Gallery object to add to the publication.

PbWizardGroup can be one of these PbWizardGroup constants.
pbWizardGroupAccentBox
pbWizardGroupAccessoryBar
pbWizardGroupAdvertisements
pbWizardGroupAttentionGetter
pbWizardGroupBarbells
pbWizardGroupBorders
pbWizardGroupBoxes
pbWizardGroupCalendars
pbWizardGroupCheckerboards
pbWizardGroupCoupon
pbWizardGroupDots
pbWizardGroupEastAsiaZipCode
pbWizardGroupJapaneseAccentBox
pbWizardGroupJapaneseAccessoryBar
pbWizardGroupJapaneseAttentionGetters
pbWizardGroupJapaneseBorders
pbWizardGroupJapaneseCalendar
pbWizardGroupJapaneseCoupons
pbWizardGroupJapaneseLinearAccent
pbWizardGroupJapaneseMarquees

pbWizardGroupJapaneseMastheads
pbWizardGroupJapanesePullQuotes
pbWizardGroupJapaneseReplyForms
pbWizardGroupJapaneseSidebars
pbWizardGroupJapaneseTableOfContents
pbWizardGroupJapaneseWebButtonEmail
pbWizardGroupJapaneseWebButtonHome
pbWizardGroupJapaneseWebButtonLink
pbWizardGroupJapaneseWebMastheads
pbWizardGroupJapaneseWebNavigationBars
pbWizardGroupJapaneseWebPullQuotes
pbWizardGroupJapaneseWebSidebars
pbWizardGroupLinearAccent
pbWizardGroupLogo
pbWizardGroupMarquee
pbWizardGroupMastheads
pbWizardGroupPhoneTearoff
pbWizardGroupPictureCaptions
pbWizardGroupPullQuotes
pbWizardGroupPunctuation
pbWizardGroupReplyForms
pbWizardGroupSidebars
pbWizardGroupTableOfContents
pbWizardGroupWebButtonsEmail
pbWizardGroupWebButtonsHome
pbWizardGroupWebButtonsLink
pbWizardGroupWebMastheads
pbWizardGroupWebNavigationBars
pbWizardGroupWebSidebars
pbWizardGroupWellPullQuotes

Left Required **Variant**. The position of the Design Gallery object's left edge relative to the left edge of the page, measured in points.

Top Required **Variant**. The position of the Design Gallery object's top edge relative to the top edge of the page, measured in points.

Width Optional **Variant**. The width of the new Design Gallery object.

Height Optional **Variant**. The height of the new Design Gallery object.

Design Optional **Long**. The design of the object to be added.

Example

This example adds a Web table of contents to the active publication.

```
ActiveDocument.Pages(1).Shapes _
    .AddGroupWizard Wizard:=pbWizardGroupTableOfContents, _
        Left:=100, Top:=100
```

AddHorizontalInVertical Method

Inserts horizontal text into a stream of vertical text and returns the new horizontal text as a **Field** object.

expression.**AddHorizontalInVertical**(***Range***, ***Text***)

expression Required. An expression that returns one of the objects in the Applies To list.

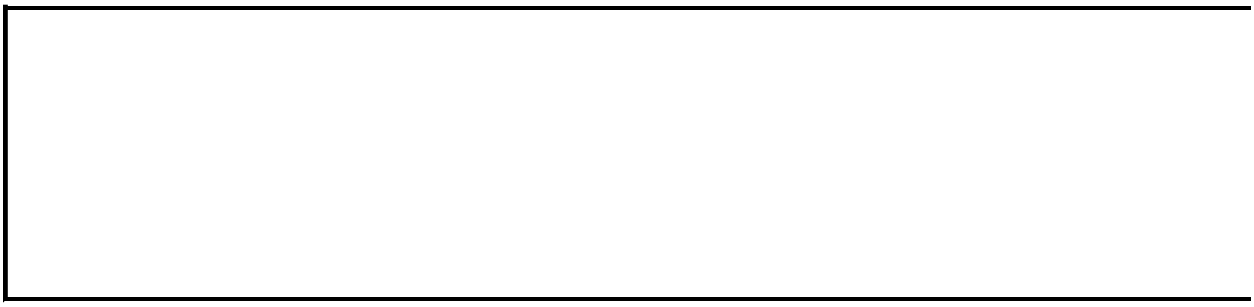
Range Required **TextRange** object. The text range at which to insert the horizontal text.

Text Required **String**. The text to be horizontally inserted.

Example

This example horizontally inserts the text "horizontal test" after the existing vertical text in shape one on page one of the active publication.

```
Dim rngTemp As TextRange
Dim fldTemp As Field

With ActiveDocument.Pages(1).Shapes(1)
    Set rngTemp = .TextFrame.TextRange.InsertAfter("")

    Set fldTemp = .TextFrame.TextRange.Fields _
        .AddHorizontalInVertical(Range:=rngTemp, Text:="horizontal t
End With
```



AddItem Method

Adds list items to a Web list box control.

expression.**AddItem**(***Item***, ***Index***, ***SelectState***, ***ItemValue***)

expression Required. An expression that returns one of the objects in the Applies To list.

Item Required **String**. The name of the item as it appears in the list.

Index Optional **Long**. The number of the list item. If ***Index*** is not specified or it is out of range of the indices of existing list box items, the new item will be added to the end of the list box. Otherwise the new item will be inserted at the position specified by ***Index*** and the index position of all items after it will be increased by one.

SelectState Optional **Boolean**. **True** if the item is selected when the list box is initially displayed. Default value is **False**.

ItemValue Optional **String**. The value of the list box item. If not specified, the new item's value will be the same as the item name.

Remarks

When you programmatically create a new Web list box, it contains three items. Use the **Delete** method to remove them from the list.

Example

This example creates a new list box control in the active publication, removes the three default list items, and then adds several items to it.

```
Sub AddListBoxItems()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlListBox, Left:=100, _
            Top:=100, Width:=150, Height:=100)
        With .WebListBox.ListBoxItems
            For intCount = 1 To .Count
                .Delete (1)
            Next
            .AddItem Item:="Green"
            .AddItem Item:="Yellow"
            .AddItem Item:="Red"
            .AddItem Item:="Blue"
            .AddItem Item:="Purple"
            .AddItem Item:="Chartreuse"
            .AddItem Item:="Pink"
            .AddItem Item:="Olive"
        End With
    End With
End Sub
```


Show All

AddLabel Method

Adds a new **Shape** object representing a text label to the specified **Shapes** collection.

expression.**AddLabel**(***Orientation***, ***Left***, ***Top***, ***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

Orientation Required **PbTextOrientation**. The orientation of the label.

PbTextOrientation can be one of these PbTextOrientation constants.

pbTextOrientationHorizontal A horizontal text label for left-to-right languages.

pbTextOrientationMixed Not used for this method.

pbTextOrientationRightToLeft A horizontal text label for right-to-left languages.

pbTextOrientationVerticalEastAsia A vertical text label for East Asian languages.

Left Required **Variant**. The position of the left edge of the shape representing the text label.

Top Required **Variant**. The position of the top edge of the shape representing the text label.

Width Required **Variant**. The width of the shape representing the text label.

Height Required **Variant**. The height of the shape representing the text label.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a new horizontal text label to the first page of the active publication.

```
Dim shpLabel As Shape

Set shpLabel = ActiveDocument.Pages(1).Shapes.AddLabel _
    (Orientation:=pbTextOrientationHorizontal, _
    Left:=144, Top:=144, _
    Width:=72, Height:=18)
```

AddLine Method

Adds a new **Shape** object representing a line to the specified **Shapes** collection.

expression.**AddLine**(***BeginX***, ***BeginY***, ***EndX***, ***EndY***)

expression Required. An expression that returns one of the objects in the Applies To list.

BeginX Required **Variant**. The x-coordinate of the beginning point of the line.

BeginY Required **Variant**. The y-coordinate of the beginning point of the line.

EndX Required **Variant**. The x-coordinate of the ending point of the line.

EndY Required **Variant**. The y-coordinate of the ending point of the line.

Remarks

For the ***BeginX***, ***BeginY***, ***EndX***, and ***EndY*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a new line to the first page of the active publication.

```
Dim shpLine As Shape

Set shpLine = ActiveDocument.Pages(1).Shapes.AddLine _
    (BeginX:=144, BeginY:=144, _
    EndX:=180, EndY:=72)
```


Show All

AddNodes Method

Inserts a new segment at the end of the freeform that's being created, and adds the nodes that define the segment. You can use this method as many times as you want to add nodes to the freeform you're creating. When you finish adding nodes, use the **ConvertToShape** method to create the freeform you've just defined.

expression.**AddNodes**(***SegmentType***, ***EditingType***, ***X1***, ***Y1***, ***X2***, ***Y2***, ***X3***, ***Y3***)

expression Required. An expression that returns one of the objects in the Applies To list.

SegmentType Required **MsoSegmentType**. The type of segment to be added.

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentCurve

msoSegmentLine

EditingType Required **MsoEditingType**. Specifies the editing type of the new node. If ***SegmentType*** is **msoSegmentLine**, ***EditingType*** must be **msoEditingAuto**; otherwise, an error occurs..

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto Adds a node type appropriate to the segments being connected.

msoEditingCorner Adds a corner node.

msoEditingSmooth Not used with this method.

msoEditingSymmetric Not used with this method.

X1 Required **Variant**. If the ***EditingType*** of the new segment is **msoEditingAuto**, this argument specifies the horizontal distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new node is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the first control point for the new segment.

Y1 Required **Variant**. If the ***EditingType*** of the new segment is **msoEditingAuto**, this argument specifies the vertical distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new node is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the first control point for the new segment.

X2 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the second control point for the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, don't specify a value for this argument.

Y2 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the second control point for the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, don't specify a value for this argument.

X3 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, don't specify a value for this argument.

Y3 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, don't specify a value for this argument.

Remarks

For the ***X1***, ***Y1***, ***X2***, ***Y2***, ***X3***, and ***Y3*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

To add nodes to a freeform after it's been created, use the **Insert** method of the **ShapeNodes** collection.

Example

This example adds a freeform with four vertices to the first page in the active publication.

```
' Add a new freeform object.
With ActiveDocument.Pages(1).Shapes _
        .BuildFreeform(EditingType:=msoEditingCorner, _
        X1:=100, Y1:=100)

    ' Add three more nodes and close the polygon.
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingCorner, _
        X1:=200, Y1:=200, X2:=225, Y2:=250, X3:=250, Y3:=200
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingAuto, X1:=200, Y1:=100
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=150, Y1:=50
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=100, Y1:=100

    ' Convert the polygon to a Shape object.
    .ConvertToShape
End With
```


Show All

AddOLEObject Method

Adds a new **Shape** object representing an OLE object to the specified **Shapes** collection.

expression.**AddOLEObject**(***Left***, ***Top***, ***Width***, ***Height***, ***ClassName***, ***FileName***, ***Link***)

expression Required. An expression that returns one of the objects in the Applies To list.

Left Required **Variant**. The position of the left edge of the shape representing the OLE object.

Top Required **Variant**. The position of the top edge of the shape representing the OLE object.

Width Optional **Variant**. The width of the shape representing the OLE object. Default is -1, meaning that the width of the shape is automatically set based on the object's data.

Height Optional **Variant**. The height of the shape representing the OLE object. Default is -1, meaning that the width of the shape is automatically set based on the object's data.

ClassName Optional **String**. The class name of the OLE object to be added.

FileName Optional **String**. The file name of the OLE object to be added. If the path isn't specified, the current working folder is used.

Link Optional **MsoTriState**. Determines whether the OLE object is linked to or embedded in the publication.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this method.

msoFalse The OLE object is embedded.

msoTriStateMixed Not used for this method.

msoTriStateToggle Not used for this method.
msoTrue *default* The OLE object is linked.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

You must specify either a ***ClassName*** or ***FileName***. If neither argument is specified or both are specified, an error occurs.

Example

The following example adds an Excel worksheet to the first page of the active publication and activates the worksheet for editing.

```
Dim shpSheet As Shape

Set shpSheet = ActiveDocument.Pages(1).Shapes.AddOLEObject _
    (Left:=72, Top:=72, ClassName:="Excel.Sheet")

shpSheet.OLEFormat.Activate
```


Show All

AddPhoneticGuide Method

Returns a **Field** object that represents phonetic text added to the specified range.

expression.**AddPhoneticGuide**(***Range***, ***Text***, ***Alignment***, ***Raise***, ***FontName***, ***FontSize***)

expression Required. An expression that returns a **Fields** object.

Range Required **TextRange** object. The text in the publication over which the phonetic text is displayed

Text Required **String**. The phonetic text to add.

Alignment Optional **PbPhoneticGuideAlignmentType**. The alignment of the added phonetic text.

PbPhoneticGuideAlignmentType can be one of these PbPhoneticGuideAlignmentType constants.

pbPhoneticGuideAlignmentCenter Centers phonetic text over the specified range.

pbPhoneticGuideAlignmentDefault *default* Centers phonetic text over the specified range.

pbPhoneticGuideAlignmentLeft Left-aligns phonetic text with the specified range.

pbPhoneticGuideAlignmentOneTwoOne Adjusts the inside and outside spacing of the phonetic text in a 1:2:1 ratio.

pbPhoneticGuideAlignmentRight Right-aligns phonetic text with the specified range.

pbPhoneticGuideAlignmentZeroOneZero Adjusts the inside and outside spacing of the phonetic text in a 0:1:0 ratio.

Raise Optional **Variant**. The distance (in points) from the top of the text in the specified range to the top of the phonetic text. If no value is specified, Publisher automatically sets the phonetic text at an optimum distance above the specified range.

FontName Optional **String**. The name of the font to use for the phonetic text. If no value is specified, Publisher uses the same font as the text in the specified range.

FontSize Optional **Variant**. The font size to use for the phonetic text. Default is 10 point.

Example

This example adds a phonetic guide to the selected phrase "very nice."

```
Sub PhoneticGuide()
    Selection.TextRange.Fields.AddPhoneticGuide _
        Range:=Selection.TextRange, Text:="ver-E nIs", _
        Alignment:=pbPhoneticGuideAlignmentCenter, _
        Raise:=11, FontSize:=7
End Sub
```


Show All

AddPicture Method

Adds a new **Shape** object representing a picture to the specified **Shapes** collection.

expression.**AddPicture**(***FileName***, ***LinkToFile***, ***SaveWithDocument***, ***Left***, ***Top***, ***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

FileName Required **String**. The name of the picture file to insert into the shape. The path can be absolute or relative.

LinkToFile Required **MsoTriState**. Determines whether the picture is linked to or embedded in the publication.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used for this method.
msoFalse The picture is to be embedded in the publication.
msoTriStateMixed Not used for this method.
msoTriStateToggle Not used for this method.
msoTrue The picture is to be linked to the publication.

SaveWithDocument Required **MsoTriState**. Determines whether the picture is saved as a separate file with the publication.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used for this method.
msoFalse The picture is embedded in the publication.
msoTriStateMixed Not used for this method.
msoTriStateToggle Not used for this method.
msoTrue A separate copy of the picture is saved as a new file in the same directory as the publication.

Left Required **Variant**. The position of the left edge of the shape representing the picture.

Top Required **Variant**. The position of the top edge of the shape representing the picture.

Width Optional **Variant**. The width of the shape representing the picture. Default is -1, meaning that the width of the shape is automatically set based on the object's data.

Height Optional **Variant**. The height of the shape representing the picture. Default is -1, meaning that the width of the shape is automatically set based on the object's data.

Remarks

If the ***SaveWithDocument*** argument is **msoTrue**, Publisher saves a new copy of the picture file specified by the ***FileName*** argument in the same directory as the publication.

The ***LinkToFile*** and ***SaveWithDocument*** arguments cannot have the same value, or else an error occurs. If either argument is **msoTrue**, the other must be **msoFalse**.

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

Example

The following example adds a picture based on an existing file to the active publication; the picture in the publication is linked to a copy of the original file. (Note that *PathToFile* must be replaced with a valid file path for this example to work.)

```
Dim shpPicture As Shape

Set shpPicture = ActiveDocument.Pages(1).Shapes.AddPicture _
    (FileName:="PathToFile", _
    LinkToFile:=msoTrue, _
    SaveWithDocument:=msoTrue
    Left:=72, Top:=72)
```

AddPolyline Method

Adds a new **Shape** object representing an open polyline or a closed polygon to the specified **Shapes** collection.

expression.**AddPolyline**(***SafeArrayOfPoints***)

expression Required. An expression that returns one of the objects in the Applies To list.

SafeArrayOfPoints Required **Variant**. An array of coordinate pairs that specifies the polyline's or polygon's vertices.

Remarks

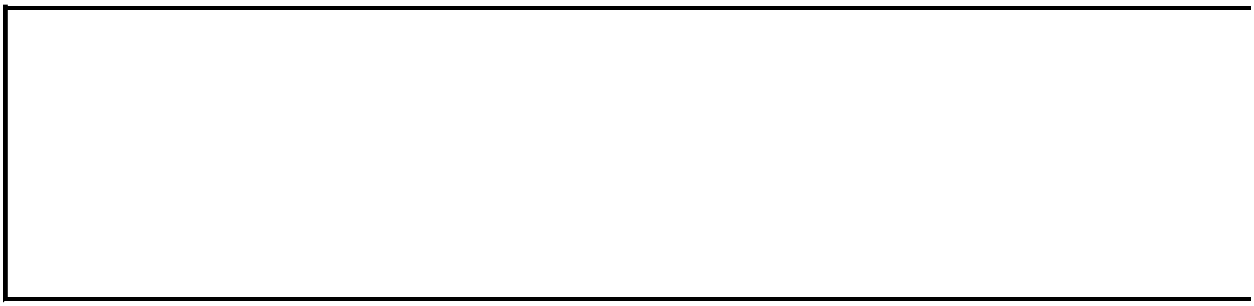
For the array elements in ***SafeArrayOfPoints***, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

To form a closed polygon, assign the same coordinates to the first and last vertices in the polyline drawing.

Example

The following example adds a triangle to the first page of the active publication. Because the first and last points have the same coordinates, the polygon is closed.

```
Dim shpPolyline As Shape
Dim arrPoints(1 To 4, 1 To 2) As Single

arrPoints(1, 1) = 25
arrPoints(1, 2) = 100
arrPoints(2, 1) = 100
arrPoints(2, 2) = 150
arrPoints(3, 1) = 150
arrPoints(3, 2) = 50
arrPoints(4, 1) = 25
arrPoints(4, 2) = 100

Set shpPolyline = ActiveDocument.Pages(1).Shapes.AddPolyline _
    (SafeArrayOfPoints:=arrPoints)
```



AddSet Method

Adds a new **WebNavigationBarSet** object representing a Web navigation bar set to the specified **WebNavigationBarSets** collection.

expression.**AddSet**(***Name***, [***Design***], [***AutoUpdate***])

expression Required. An expression that returns a **WebNavigationBarSet** object.

Name Required **String**. The name of the Web navigation bar to be added. This parameter must be unique.

Design Optional **pbWizardNavBarDesign**. Specifies the navigation bar design scheme.

AutoUpdate Optional **Boolean**. **True** if all pages with the **AddHyperlinkToWebNavBar** property set to **True** are added as links to the navigation bar and the navigation bar is kept updated.

Remarks

The **Name** parameter must be unique to avoid a run time error.

Example

The following example adds a **WebNavigationBarSet** object to the **WebNavigationBarSets** collection of the active document then sets some properties.

```
Dim objWebNavBarSet As WebNavigationBarSet

Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets.AddSet( _
    Name:="WebNavBarSet1", _
    Design:=pbnbDesignAmbient, _
    AutoUpdate:=True)

With objWebNavBarSet
    .AddToEveryPage Left:=50, Top:=10
    .ButtonStyle = pbnbDesignTopLine
    .ChangeOrientation pbNavBarOrientHorizontal
End With
```

AddShape Method

Adds a new **Shape** object representing an AutoShape to the specified **Shapes** collection.

expression.**AddShape**(***Type***, ***Left***, ***Top***, ***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

Type Required **MsoAutoShapeType**. The type of AutoShape to draw. For a complete list of **MsoAutoShapeType** constants, see the Object Browser.

Left Required **Variant**. The position of the left edge of the shape representing the AutoShape.

Top Required **Variant**. The position of the top edge of the shape representing the AutoShape.

Width Required **Variant**. The width of the shape representing the AutoShape.

Height Required **Variant**. The height of the shape representing the AutoShape.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a rectangle to the first page of the active publication.

```
Dim shpShape As Shape

Set shpShape = ActiveDocument.Pages(1).Shapes.AddShape _
    (Type:=msoShapeRectangle, _
    Left:=144, Top:=144, _
    Width:=72, Height:=144)
```


Show All

AddTable Method

Adds a new **Shape** object representing a table to the specified **Shapes** collection.

expression.**AddTable**(***NumRows***, ***NumColumns***, ***Left***, ***Top***, ***Width***, ***Height***, ***FixedSize***, ***Direction***)

expression Required. An expression that returns one of the objects in the Applies To list.

NumRows Required **Long**. The number of rows in the new table. Values between 1 and 128 are valid; any values outside this range will generate an error.

NumColumns Required **Long**. The number of columns in the new table. Values between 1 and 128 are valid; any values outside this range will generate an error.

Left Required **Variant**. The position of the left edge of the shape representing the table.

Top Required **Variant**. The position of the top edge of the shape representing the table.

Width Required **Variant**. The width of the shape representing the table.

Height Required **Variant**. The height of the shape representing the table.

FixedSize Optional **Boolean**. **True** if Microsoft Publisher reduces the number of rows and columns of the table to fit the specified width and height. **False** if Microsoft Publisher automatically increases the width and height of the table frame to accommodate the number of rows and columns in the table. Default is **False**.

Direction Optional **PbTableDirectionType**. The direction in which table columns are numbered. The default depends on the current language setting.

PbTableDirectionType can be one of these PbTableDirectionType constants.

pbTableDirectionLeftToRight Table columns are numbered from left to right. Default for left-to-right languages.

pbTableDirectionRightToLeft Table columns are numbered from right to left. Default for right-to-left languages.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

This example creates a new table on the first page of the active publication.

```
Dim shpTable As Shape

Set shpTable = ActiveDocument.Pages(1).Shapes.AddTable _
    (NumRows:=3, NumColumns:=4, _
    Left:=10, Top:=10, _
    Width:=288, Height:=216)
```


Show All

AddTextbox Method

Adds a new **Shape** object representing a text box to the specified **Shapes** collection.

expression.**AddTextbox**(***Orientation***, ***Left***, ***Top***, ***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

Orientation Required **PbTextOrientation**. The orientation of the text box.

PbTextOrientation can be one of these PbTextOrientation constants.

pbTextOrientationHorizontal A horizontal text box for left-to-right languages.

pbTextOrientationMixed Not used for this method.

pbTextOrientationRightToLeft A horizontal text box for right-to-left languages. This value has no effect if a right-to-left language is not selected.

pbTextOrientationVerticalEastAsia A vertical text box for East Asian languages. If a non-East Asian language is selected, text appears rotated 90 degrees to the right.

Left Required **Variant**. The position of the left edge of the shape representing the text box.

Top Required **Variant**. The position of the top edge of the shape representing the text box.

Width Required **Variant**. The width of the shape representing the text box.

Height Required **Variant**. The height of the shape representing the text box.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

The following example adds a new horizontal text box to the first page of the active publication.

```
Dim shpTextBox As Shape

Set shpTextBox = ActiveDocument.Pages(1).Shapes.AddTextBox _
    (Orientation:=pbTextOrientationHorizontal, _
    Left:=144, Top:=144, _
    Width:=72, Height:=18)
```


Show All

AddTextEffect Method

Adds a new **Shape** object representing a WordArt object to the specified **Shapes** collection.

expression.**AddTextEffect**(***PresetTextEffect***, ***Text***, ***FontName***, ***FontSize***, ***FontBold***, ***FontItalic***, ***Left***, ***Top***)

expression Required. An expression that returns one of the objects in the Applies To list.

PresetTextEffect Required **MsoPresetTextEffect**. The preset text effect to use. The values of the **MsoPresetTextEffect** constants correspond to the formats listed in the **WordArt Gallery** dialog box (numbered from left to right and from top to bottom).

MsoPresetTextEffect can be one of these MsoPresetTextEffect constants.

msoTextEffect1
msoTextEffect2
msoTextEffect3
msoTextEffect4
msoTextEffect5
msoTextEffect6
msoTextEffect7
msoTextEffect8
msoTextEffect9
msoTextEffect10
msoTextEffect11
msoTextEffect12
msoTextEffect13
msoTextEffect14
msoTextEffect15
msoTextEffect16
msoTextEffect17

msoTextEffect18
msoTextEffect19
msoTextEffect20
msoTextEffect21
msoTextEffect22
msoTextEffect23
msoTextEffect24
msoTextEffect25
msoTextEffect26
msoTextEffect27
msoTextEffect28
msoTextEffect29
msoTextEffect30
msoTextEffectMixed Not used for this method.

Text Required **String**. The text to use for the WordArt object.

FontName Required **String**. The name of the font to use for the WordArt object.

FontSize Required **Variant**. The font size to use for the WordArt object. Numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

FontBold Required **MsoTriState**. Determines whether to format the WordArt text as bold.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this method.
msoFalse Do not format the WordArt text as bold.
msoTriStateMixed Not used with this method.
msoTriStateToggle Not used with this method.
msoTrue Format the WordArt text as bold.

FontItalic Required **MsoTriState**. Determines whether to format the WordArt text as italic.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this method.

msoFalse Do not format the WordArt text as italic.

msoTriStateMixed Not used with this method.

msoTriStateToggle Not used with this method.

msoTrue Format the WordArt text as italic.

Left Required **Variant**. The position of the left edge of the shape representing the WordArt object.

Top Required **Variant**. The position of the top edge of the shape representing the WordArt object.

Remarks

For the ***Left*** and ***Top*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

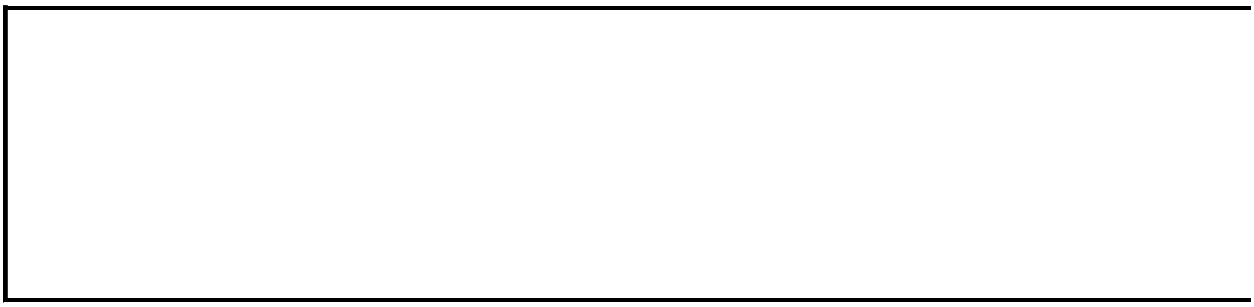
The height and width of the WordArt object is determined by its text and formatting.

Use the **TextEffect** property to return a **TextEffectFormat** object whose properties can be used to edit an existing WordArt object.

Example

The following example adds a WordArt object to the first page of the active publication.

```
Dim shpWordArt As Shape

Set shpWordArt = ActiveDocument.Pages(1).Shapes.AddTextEffect _
    (PresetTextEffect:=msoTextEffect7, Text:="Annual Report", _
    FontName:="Arial Black", FontSize:=24, _
    FontBold:=msoFalse, FontItalic:=msoFalse, _
    Left:=144, Top:=72)
```




Show All

AddToCatalogMergeArea Method

Adds the specified shape or shapes to the publication page's catalog merge area.

expression.**AddToCatalogMergeArea**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The catalog merge area is automatically resized to accommodate objects that are larger than the merge area, or that are positioned outside the catalog merge area when they are added.

The **AddToCatalogMergeArea** method does not apply to merge data fields:

- Use the **Insert** method of the **MailMergeDataFields** collection to add a picture data field to a publication page's catalog merge area.
- Use the **InsertMailMergeField** method of the **TextRange** object to add a text data field to a text box.

Note that to add a text box that will contain text data fields to a catalog merge area, you use the **AddToCatalogMergeArea** method.

Example

The following example adds a rectangle to the catalog merge area on the first page of the specified publication. This example assumes a catalog merge area has been added to the first page.

```
ThisDocument.Pages(1).Shapes.AddShape(1, 80, 75, 450, 125).AddToCata
```

AddToEveryPage Method

Adds a **ShapeRange** of type **pbWebNavigationBar** to each page of the current document.

expression.**AddToEveryPage**(***Left***, ***Top***, [***Width***])

expression Required. An expression that returns a **WebNavigationBarSet** object.

Left Required **Variant**. The position of the left edge of the shape representing the Web navigation bar set.

Top Required **Variant**. The position of the top edge of the shape representing the Web navigation bar set.

Width Optional **Variant**. The width of the shape representing the Web navigation bar set.

Remarks

The specified Web navigation bar set must exist before calling this method.

Example

The following example adds a Web navigation bar set named "WebNavBarSet1" to the top of every page in the active document.

```
ActiveDocument.WebNavigationBarSets("WebNavBarSet1") _
    .AddToEveryPage Left:=10, Top:=20
```

The following example adds a new Web navigation bar set to the active document and adds it to every page of the publication.

```
Dim objWebNavBarSet As WebNavigationBarSet

Set objWebNavBarSet = ActiveDocument.WebNavigationBarSets.AddSet( _
    Name:="WebNavBarSet1", _
    Design:=pbnbDesignTopLine, _
    AutoUpdate:=True)

objWebNavBarSet.AddToEveryPage Left:=50, Top:=10, Width:=500
```


Show All

AddWebControl Method

Adds a new **Shape** object representing a Web form control to the specified **Shapes** collection.

expression.**AddWebControl**(***Type***, ***Left***, ***Top***, ***Width***, ***Height***, ***LaunchPropertiesWindow***)

expression Required. An expression that returns one of the objects in the Applies To list.

Type Required **PbWebControlType**. Specifies the type of Web form control to add. An error occurs if **pbWebControlWebComponent** is used.

PbWebControlType can be one of these PbWebControlType constants.
pbWebControlCheckBox Adds a check box.
pbWebControlCommandButton Adds a command button.
pbWebControlHotSpot Adds a hot spot.
pbWebControlHTMLFragment Adds an HTML fragment.
pbWebControlListBox Adds a list box.
pbWebControlMultiLineTextBox Adds a multiple-line text area.
pbWebControlOptionButton Adds an option button.
pbWebControlSingleLineTextBox Adds a single-line text box.
pbWebControlWebComponent Not used for this method.

Left Required **Variant**. The position of the left edge of the shape representing the Web form control.

Top Required **Variant**. The position of the top edge of the shape representing the Web form control.

Width Required **Variant**. The width of the shape representing the Web form control. For command buttons, this parameter is ignored.

Height Required **Variant**. The height of the shape representing the Web form

control. For command buttons, this parameter is ignored.

LaunchPropertiesWindow Optional **Boolean**. Not supported. Default is **False**; an error occurs if this argument is set to **True**.

Remarks

For the ***Left***, ***Top***, ***Width***, and ***Height*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

When adding a hot spot to a Web control by using the **pbWebControlHotSpot** constant, the URL is specified by the **Hyperlink** property.

Note that the **Shape.Fill** property, which returns a **FillFormat** object, and the **Shape.Line** property, which returns a **LineFormat** object, cannot be accessed from a hot spot shape. A run-time error is returned if attempting to access these properties from a hot spot shape.

Example

The following example adds a Web form check box control to the first page of the active publication.

```
Dim shpCheckBox As Shape

Set shpCheckBox = ActiveDocument.Pages(1).Shapes.AddWebControl _
    (Type:=pbWebControlCheckBox, _
    Left:=216, Top:=216, _
    Width:=18, Height:=18)
```

The following example adds hot spots to a shape on page four of the active Web publication. First, a four-point star AutoShape is added to the page. Next, a hot spot is added to each arm of the star by using the **AddWebControl** method with a Type of **pbWebControlHotSpot**. Finally, a hyperlink is added to each hot spot by using the **Hyperlink** property of each hot spot shape.

```
Dim theDoc As Document
Dim theStar As Shape
Dim theWC1 As Shape
Dim theWC2 As Shape
Dim theWC3 As Shape
Dim theWC4 As Shape

Set theDoc = ActiveDocument
Set theStar = theDoc.Pages(4).Shapes.AddShape _
        (Type:=msoShape4pointStar, Left:=200, Top:=25, _
        Width:=200, Height:=200)

With theDoc.Pages(4).Shapes

    Set theWC1 = .addWebControl(Type:=pbWebControlHotSpot, _
        Left:=280, Top:=25, Width:=40, Height:=80)
        With theWC1
            .Hyperlink.Address = "http://www.contoso.com/page1.htm"
        End With

    Set theWC2 = .addWebControl(Type:=pbWebControlHotSpot, _
        Left:=320, Top:=105, Width:=80, Height:=40)
        With theWC2
            .Hyperlink.Address = "http://www.contoso.com/page2.htm"
        End With
```

```
    Set theWC3 = .addWebControl(Type:=pbWebControlHotSpot, _
        Left:=280, Top:=145, Width:=40, Height:=80)
        With theWC3
            .Hyperlink.Address = "http://www.contoso.com/page3.htm"
        End With

    Set theWC4 = .addWebControl(Type:=pbWebControlHotSpot, _
        Left:=200, Top:=105, Width:=80, Height:=40)
        With theWC4
            .Hyperlink.Address = "http://www.contoso.com/page4.htm"
        End With
End With
```

AddWebNavigationBar Method

Adds a **Shape** object of type **pbWebNavigationBar** to the current page of a publication.

expression.**AddWebNavigationBar**(***Name***, ***Left***, ***Top***, [***Width***])

expression Required. An expression that returns a **Shape** object.

Name Required **String**. The name of the **WebNavigationBarSet** object to be added to the specified **Shape**.

Left Required **Variant**. The position of the left edge of the shape representing the Web navigation bar set.

Top Required **Variant**. The position of the top edge of the shape representing the Web navigation bar set.

Width Optional **Variant**. The width of the shape representing the Web navigation bar set.

Remarks

The AddWebNavigationBar method does not create a new Web navigation bar set. It adds an existing set from the **WebNavigationBarSets** collection with the name passed in as the **Name** parameter.

Example

The following example adds a **WebNavigationBarSet** to the active document.

```
Dim shpShape As Shape
ActiveDocument.WebNavigationBarSets.AddSet Name:="NavBar", AutoUpdat
Set shpShape = ActiveDocument.Pages(1).Shapes.AddWebNavigationBar _
    (Name:="NavBar", Left:=10, Top:=25)
```


Show All

AddWizardPage Method

Adds the specified new wizard page to a specified location in a publication.

expression.**AddWizardPage**(***After***, ***PageType***, [***AddHyperLinkToWebNavBar***])

expression Required. An expression that returns one of the objects in the Applies To list.

After Required **Long**. The page after which to place the new wizard page.

PageType Optional **PbWizardPageType**. The type of wizard page to add.

AddHyperLinkToWebNavBar Optional **Boolean**. Specifies whether a link to the new page will be added to the automatic navigation bars of existing pages. Default is **False**, which means that if this argument is omitted, links to this page will not be added to the automatic navigation bars of existing pages.

PbWizardPageType can be one of these PbWizardPageType constants.
pbWizardPageTypeCatalogBlank
pbWizardPageTypeCatalogCalendar
pbWizardPageTypeCatalogEightItemsOneColumn
pbWizardPageTypeCatalogEightItemsTwoColumns
pbWizardPageTypeCatalogFeaturedItem
pbWizardPageTypeCatalogForm
pbWizardPageTypeCatalogFourItemsAlignedPictures
pbWizardPageTypeCatalogFourItemsOffsetPictures
pbWizardPageTypeCatalogFourItemsSquaredPictures
pbWizardPageTypeCatalogOneColumnText
pbWizardPageTypeCatalogOneColumnTextPicture
pbWizardPageTypeCatalogTableOfContents
pbWizardPageTypeCatalogThreeItemsAlignedPictures
pbWizardPageTypeCatalogThreeItemsOffsetPictures
pbWizardPageTypeCatalogThreeItemsStackedPictures

pbWizardPageTypeCatalogTwoColumnsText
pbWizardPageTypeCatalogTwoColumnsTextPicture
pbWizardPageTypeCatalogTwoItemsAlignedPictures
pbWizardPageTypeCatalogTwoItemsOffsetPictures
pbWizardPageTypeNewsletter3Stories
pbWizardPageTypeNewsletterCalendar
pbWizardPageTypeNewsletterOrderForm
pbWizardPageTypeNewsletterResponseForm
pbWizardPageTypeNewsletterSignupForm
pbWizardPageTypeNone *default*
pbWizardPageTypeWebCalendar
pbWizardPageTypeWebEvent
pbWizardPageTypeWebPriceList
pbWizardPageTypeWebRelatedLinks
pbWizardPageTypeWebSpecialOffer
pbWizardPageTypeWebStory

Remarks

You can only add wizard pages to similar wizard publications. For example, you can add a Catalog Calendar Wizard page to a catalog but not to a newsletter. An error occurs if you try to add a wizard page to a different type of publication.

Example

This example creates a new catalog publication, adds the wizard calendar page after the first page of the catalog, and adds the page as a link to each Web navigation bar set of the publication.

```
Sub AddNewWizardPage()
    Dim PubApp As Publisher.Application
    Dim PubDoc As Publisher.Document
    Set PubApp = New Publisher.Application
    Set PubDoc = PubApp.NewDocument(Wizard:=pbWizardCatalogs, _
        Design:=7)
    PubDoc.Pages.AddWizardPage After:=1, _
        PageType:=pbWizardPageTypeCatalogCalendar, _
        AddHyperLinkToWebNavBar:=True
    PubApp.ActiveWindow.Visible = True
End Sub
```

This example verifies that the active document is a catalog and, if it is, adds a catalog form after the first page but does not add the page as a link in any Web navigation bar sets.

```
Sub InsertCatalogWizardPage()
    With ActiveDocument
        If .Wizard.ID = 161 Then
            .Pages.AddWizardPage After:=1, _
                PageType:=pbWizardPageTypeCatalogForm, _
                AddHyperLinkToWebNavBar:=False
        End If
    End With
End Sub
```


Show All

Align Method

Aligns all the shapes in the specified **ShapeRange** object.

expression.**Align**(***AlignCmd***, ***RelativeTo***)

expression Required. An expression that returns one of the objects in the Applies To list.

AlignCmd Required **MsoAlignCmd**. Specifies how the shapes are to be aligned.

MsoAlignCmd can be one of these MsoAlignCmd constants.

msoAlignBottoms Aligns shapes along their bottom edges. If ***RelativeTo*** is **msoFalse**, the bottommost shape determines the line against which the other shapes are aligned.

msoAlignCenters Aligns shapes on a vertical line through their centers. If ***RelativeTo*** is **msoFalse**, shapes are aligned on a line halfway between the left- and rightmost shapes.

msoAlignLefts Aligns shapes along their left edges. If ***RelativeTo*** is **msoFalse**, the leftmost shape determines the line against which the other shapes are aligned.

msoAlignMiddles Aligns shapes on a horizontal line through their centers. If ***RelativeTo*** is **msoFalse**, shapes are aligned on a line halfway between the top- and bottommost shapes.

msoAlignRights Aligns shapes along their right edges. If ***RelativeTo*** is **msoFalse**, the rightmost shape determines the line against which the other shapes are aligned.

msoAlignTops Aligns shapes along their top edges. If ***RelativeTo*** is **msoFalse**, the topmost shape determines the line against which the other shapes are aligned.

RelativeTo Required **MsoTriState**. Specifies whether shapes are aligned relative to the page or to one another.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this method.
msoFalse Aligns shapes relative to one another.
msoTriStateMixed Not used with this method.
msoTriStateToggle Not used with this method.
msoTrue Aligns shapes relative to the page.

Remarks

If the ***RelativeTo*** argument is **msoFalse** and the shape range contains only one shape, an error occurs.

Example

The following example aligns all the shapes on the first page of the active publication on a vertical line through their centers.

```
ActiveDocument.Pages(1).Shapes.Range.Align _
    AlignCmd:=msoAlignCenters, _
    RelativeTo:=msoTrue
```

Apply Method

Applies formatting copied from another shape or shape range using the **PickUp** method.

expression.**Apply**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If you do not first use the **PickUp** method to copy the formatting from another shape, an error occurs.

Example

The following example copies the formatting from the first shape of the active publication to the second shape of the active publication.

```
With ActiveDocument.Pages(1)
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```


Show All

ApplyAutoFormat Method

Applies automatic built-in table formatting to a specified table.

expression.**ApplyAutoFormat**(***AutoFormat***, ***TextFormatting***, ***TextAlignment***, ***Fill***, ***Borders***)

expression Required. An expression that returns one of the objects in the Applies To list.

AutoFormat Required **PbTableAutoFormatType**. The type of automatic formatting to apply to the specified table.

PbTableAutoFormatType can be one of these PbTableAutoFormatType constants.

pbTableAutoFormatCheckbookRegister
pbTableAutoFormatCheckerboard
pbTableAutoFormatDefault
pbTableAutoFormatList1
pbTableAutoFormatList2
pbTableAutoFormatList3
pbTableAutoFormatList4
pbTableAutoFormatList5
pbTableAutoFormatList6
pbTableAutoFormatList7
pbTableAutoFormatListWithTitle1
pbTableAutoFormatListWithTitle2
pbTableAutoFormatListWithTitle3
pbTableAutoFormatMixed
pbTableAutoFormatNone
pbTableAutoFormatNumbers1
pbTableAutoFormatNumbers2
pbTableAutoFormatNumbers3

pbTableAutoFormatNumbers4
pbTableAutoFormatNumbers5
pbTableAutoFormatNumbers6
pbTableAutoFormatTableOfContents1
pbTableAutoFormatTableOfContents2
pbTableAutoFormatTableOfContents3

TextFormatting Optional **Boolean**. **True** to apply font formatting to the text in the table. Default value is **True**.

TextAlignment Optional **Boolean**. **True** to apply text alignment to the text in the table. Default value is **True**.

Fill Optional **Boolean**. **True** to apply fill formatting to cells in the table. Default value is **True**.

Borders Optional **Boolean**. **True** to apply borders to cells in the table. Default value is **True**.

Example

This example applies the checkbook register automatic formatting, with fill and borders, to the specified table.

```
Sub ApplyAutomaticTableFormatting()
    ActiveDocument.Pages(1).Shapes(1).Table.ApplyAutoFormat _
        AutoFormat:=pbTableAutoFormatCheckbookRegister, _
        Borders:=False
End Sub
```

ApplyCustomDropCap Method

Applies custom formatting to the first letters of paragraphs in a text frame.

expression.**ApplyCustomDropCap**(***LinesUp***, ***Size***, ***Span***, ***FontName***, ***Bold***, ***Italic***)

expression Required. An expression that returns one of the objects in the Applies To list.

LinesUp Optional **Long**. The number of lines to move up the drop cap. The default is 0. The maximum number cannot be more than the number entered for the ***Size*** argument less one.

Size Optional **Long**. The size of the drop cap letters in number of lines high. The default is 5.

Span Optional **Long**. The number of letters included in the drop cap. The default is 1.

FontName Optional **String**. The name of the font to format the drop cap. The default is the current font.

Bold Optional **Boolean**. **True** to bold the drop cap. The default is **False**.

Italic Optional **Boolean**. **True** to italicize the drop cap. The default is **False**.

Example

This example formats the first three letters of the paragraphs in the specified text box.

```
Sub CustDropCap()

    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange.DropCap _
        .ApplyCustomDropCap LinesUp:=1, Size:=6, Span:=3, _
        FontName:="Script MT Bold", Bold:=True, Italic:=True

End Sub
```

ApplyFilter Method

Applies a filter to a mail merge data source to remove (or filter out) specified records containing (or not containing) specific data.

expression.**ApplyFilter**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a new filter that removes all records with a blank Region field and then applies the filter to the active publication. This example assumes that a mail merge data source is attached to the active publication.

```
Sub FilterDataSource()
    With ActiveDocument.MailMerge.DataSource
        .Filters.Add Column:="Region", _
            Comparison:=msoFilterComparisonIsBlank, _
            Conjunction:=msoFilterConjunctionAnd
        .ApplyFilter
    End With
End Sub
```

AutomaticLength Method

Specifies that the first segment of the callout line (the segment attached to the text callout box) be scaled automatically when the callout is moved.

expression.**AutomaticLength**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Calling this method sets the **AutoLength** property of the specified object to **msoTrue**.

Use the **CustomLength** method to specify that the first segment of the callout line retain the fixed length returned by the **Length** property whenever the callout is moved. Applies only to callouts whose lines consist of more than one segment (types **msoCalloutThree** and **msoCalloutFour**).

Example

This example toggles between an automatically-scaling first segment and one with a fixed length for the callout line for the first shape in the active publication. For the example to work, this shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    If .AutoLength Then
        .CustomLength Length:=50
    Else
        .AutomaticLength
    End If
End With
```

BeginConnect Method

Attaches the beginning of the specified connector to a specified shape.

expression.**BeginConnect**(***ConnectedShape***, ***ConnectionSite***)

expression Required. An expression that returns one of the objects in the Applies To list.

ConnectedShape Required **Shape** object. The shape to which Microsoft Publisher attaches the beginning of the connector. The specified **Shape** object must be in the same **Shapes** collection as the connector.

ConnectionSite Required **Long**. A connection site on the shape specified by ***ConnectedShape***. Must be an integer between 1 and the integer returned by the **ConnectionSiteCount** property of the specified shape. Connection sites are numbered starting from the top of the specified shape and moving counterclockwise around the shape. If you want the connector to automatically find the shortest path between the two shapes it connects, specify any valid integer for this argument and then use the **RerouteConnections** method after the connector is attached to shapes at both ends.

Remarks

If there's already a connection between the beginning of the connector and another shape, that connection is broken. If the beginning of the connector isn't already positioned at the specified connecting site, this method moves the beginning of the connector to the connecting site and adjusts the size and position of the connector.

When you attach a connector to an object, the size and position of the connector are automatically adjusted if necessary.

Use the **EndConnect** method to attach the end of the connector to a shape.

Example

This example adds two rectangles to the first page in the active publication and connects them with a curved connector. Note that the **RerouteConnections** method overrides the values you supply for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new curved connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections
    End With

End With
```

BeginCustomUndoAction Method

Specifies the starting point and label (textual description) of a group of actions that are wrapped to create a single undo action. The **EndCustomUndoAction** method is used to specify the end point of the actions used to create the single undo action. The wrapped group of actions can be undone with a single undo.

expression.**BeginCustomUndoAction**(***ActionName***)

expression Required. An expression that returns a **Document** object.

ActionName Required **String**. The label that corresponds to the single undo action. This label appears when you click the arrow beside the **Undo** button on the **Standard** toolbar.

Remarks

The following methods of the **Document** object are disabled within a custom undo action. A run-time error is returned if any of these methods are called within a custom undo action:

- **Document.Close**
- **Document.MailMerge.DataSource.Close**
- **Document.PrintOut**
- **Document.Redo**
- **Document.Save**
- **Document.SaveAs**
- **Document.Undo**
- **Document.UndoClear**
- **Document.UpdateOLEObjects**

The **BeginCustomUndoAction** method must be called before the **EndCustomUndoAction** method is called. A run-time error is returned if **EndCustomUndoAction** is called before **BeginCustomUndoAction**.

Nesting a custom undo action within another custom undo action is allowed, but the nested custom undo action will have no effect. Only the outermost custom undo action will be active.

Example

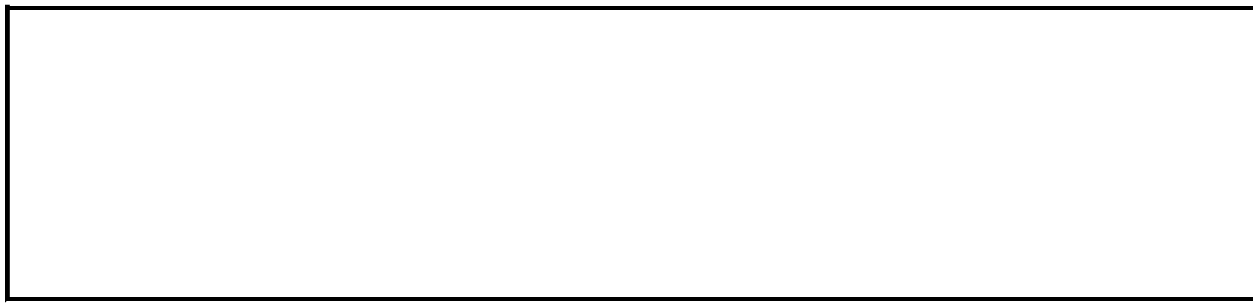
The following example contains two custom undo actions. The first one is created on the first page of the active publication. The **BeginCustomUndoAction** method is used to specify the point at which the custom undo action should begin. Six individual actions are performed, and then they are wrapped into one action with the call to **EndCustomUndoAction**.

The text in the text frame that was created within the first custom undo action is then tested to determine whether the font is Verdana. If not, the **Undo** method is called with **UndoActionsAvailable** passed as a parameter. In this case there is only one undo action available. So, the call to **Undo** will only undo one action, but this one action has wrapped six actions into one.

A second undo action is then created, and it could also be undone later with a single undo operation.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(1)

With theDoc
    ' The following six actions are wrapped to create one
    ' custom undo action named "Add Rectangle and Courier Text".
    .BeginCustomUndoAction ("Add Rectangle and Courier Text")
    With thePage
        Set theShape = .Shapes.AddShape(msoShapeRectangle, _
            75, 75, 190, 30)
        With theShape.TextFrame.TextRange
             .Font.Size = 14
             .Font.Bold = msoTrue
             .Font.Name = "Courier"
             .Text = "This font is Courier."
        End With
     End With
     .EndCustomUndoAction

    If Not thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Verdan
        ' This call to Undo will undo all actions that are available
        ' In this case, there is only one action that can be undone.
```

```
        .Undo (.UndoActionsAvailable)
        ' A new custom undo action is created with a name of
        ' "Add Balloon and Verdana Text".
        .BeginCustomUndoAction ("Add Balloon and Verdana Text")
        With thePage
            Set theShape = .Shapes.AddShape(msoShapeBalloon, _
                75, 75, 190, 30)
            With theShape.TextFrame.TextRange
                .Font.Size = 11
                .Font.Name = "Verdana"
                .Text = "This font is Verdana."
            End With
        End With
        .EndCustomUndoAction
    End If
End With
```



BeginDisconnect Method

Detaches the beginning of the specified connector from the shape to which it's attached.

expression.**BeginDisconnect**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This method doesn't alter the size or position of the connector: the beginning of the connector remains positioned at a connection site but is no longer connected.

Use the **EndDisconnect** method to detach the end of the connector from a shape.

Example

This example adds two rectangles to the first page in the active publication, attaches them with a connector, automatically reroutes the connector along the shortest path, and then detaches the connector from the rectangles.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=0, EndY:=0) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections

        ' Disconnect the new connector from the rectangles but
        ' leave in place.
        .BeginDisconnect
        .EndDisconnect
    End With

End With
```

BreakForwardLink Method

Breaks the forward link for the specified text frame, if such a link exists.

expression.**BreakForwardLink**

expression Required. An expression that returns a **TextFrame** object.

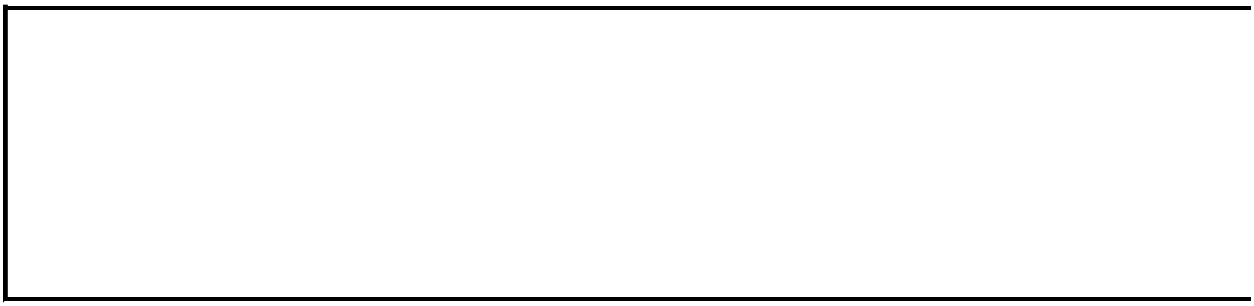
Remarks

Applying this method to a shape in the middle of a chain of shapes with linked text frames will break the chain, leaving two sets of linked shapes. All of the text, however, will remain in the first series of linked shapes.

Example

This example creates a new publication, adds a chain of three linked text boxes to it, and then breaks the link after the second text box.

```
Sub BreakTextLink()
    Dim shpTextbox1 As Shape
    Dim shpTextbox2 As Shape
    Dim shpTextbox3 As Shape

    Set shpTextbox1 = ActiveDocument.Pages(1).Shapes.AddTextbox _
        (Orientation:=msoTextOrientationHorizontal, _
        Left:=72, Top:=36, Width:=72, Height:=36)
    shpTextbox1.TextFrame.TextRange = "This is some text. " _
        & "This is some more text. This is even more text. " _
        & "And this is some more text and even more text."

    Set shpTextbox2 = ActiveDocument.Pages(1).Shapes.AddTextbox _
        (Orientation:=msoTextOrientationHorizontal, _
        Left:=72, Top:=108, Width:=72, Height:=36)

    Set shpTextbox3 = ActiveDocument.Pages(1).Shapes.AddTextbox _
        (Orientation:=msoTextOrientationHorizontal, _
        Left:=72, Top:=180, Width:=72, Height:=36)

    shpTextbox1.TextFrame.NextLinkedTextFrame = shpTextbox2.TextFram
    shpTextbox2.TextFrame.NextLinkedTextFrame = shpTextbox3.TextFram
    MsgBox "Textboxes 1, 2, and 3 are linked."
    shpTextbox2.TextFrame.BreakForwardLink
End Sub
```




Show All

BuildFreeform Method

Builds a freeform object. Returns a **FreeformBuilder** object that represents the freeform as it is being built.

expression.**BuildFreeform**(***EditingType***, ***X1***, ***Y1***)

expression Required. An expression that returns one of the objects in the Applies To list.

EditingType Required **MsoEditingType**. Specifies the editing type of the first node.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto Adds a node type appropriate to the segments being connected.

msoEditingCorner Adds a corner node.

msoEditingSmooth Not used with this method.

msoEditingSymmetric Not used with this method.

X1 Required **Variant**. The horizontal position of the first node in the freeform drawing relative to the upper-left corner of the page.

Y1 Required **Variant**. The vertical position of the first node in the freeform drawing relative to the upper-left corner of the page.

Remarks

For the ***X1*** and ***Y1*** arguments, numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Use the **AddNodes** method to add segments to the freeform. After you have added at least one segment to the freeform, you can use the **ConvertToShape** method to convert the **FreeformBuilder** object into a **Shape** object that has the geometric description you've defined in the **FreeformBuilder** object.

Example

This example adds a freeform with four segments to the first page of the active publication.

```
' Add a new freeform object.
With ActiveDocument.Shapes _
        .BuildFreeform(EditingType:=msoEditingCorner, _
        X1:=100, Y1:=100)

    ' Add three more nodes and close the polygon.
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingCorner, _
        X1:=200, Y1:=200, X2:=225, Y2:=250, X3:=250, Y3:=200
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingAuto, X1:=200, Y1:=100
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=150, Y1:=50
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=100, Y1:=100

    ' Convert the polygon to a Shape object.
    .ConvertToShape
End With
```

CentimetersToPoints Method

Converts a measurement from centimeters to points (1 cm = 28.35 points). Returns the converted measurement as a **Single**.

expression.**CentimetersToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The centimeter value to be converted to points.

Remarks

Use the **PointsToCentimeters** method to convert measurements in points to centimeters.

Example

This example converts measurements in centimeters entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in centimeters (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " cm = " _
        & Format(Application _
        .CentimetersToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```

ChangeFileOpenDirectory Method

Sets the folder in which Publisher searches for documents. The specified folder's contents are listed the next time the **Open Publication** dialog box (**File** menu) is displayed.

expression.**ChangeFileOpenDirectory**(***Dir***)

expression Required. An expression that returns one of the objects in the Applies To list.

Dir Required **String**. The directory path.

Remarks

Publisher searches the specified folder for documents until the user changes the folder in the **Open Publication** dialog box or the current Publisher session ends. Use the **PathForPublications** property of the **Options** object to change the default folder for documents in every Publisher session.

Example

This example changes the folder in which Publisher searches for documents. (Note that *PathToDirectory* must be replaced with a valid file path for this example to work.)

```
Sub ChangeOpenPath()
    ChangeFileOpenDirectory Dir:="PathToDirectory"
End Sub
```


Show All

ChangeOrientation Method

Sets a **PbNavBarOrientation** constant that represents the alignment of the navigation bar; vertical or horizontal.

ChangeOrientation can be set to one of these **PbNavBarOrientation** constants:

pbNavBarOrientHorizontal

pbNavBarOrientVertical

expression.**ChangeOrientation**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Example

The following example sets an object variable to the first Web navigation bar set in the active document, adds it every page, changes the orientation to horizontal, sets the horizontal alignment to center, and then sets the horizontal button count to 4.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets(1)
With objWebNav
    .AddToEveryPage Left:=10, Top:=10
    .ChangeOrientation pbNavBarOrientHorizontal
    .HorizontalAlignment = pbnbAlignCenter
    .HorizontalButtonCount = 4
End With
```

Characters Method

Returns a **TextRange** object that represents the specified subset of text characters.

expression.**Characters**(***Start***, ***Length***)

expression Required. An expression that returns one of the objects in the Applies To list.

Start Required **Long**. The first character in the returned range.

Length Optional **Long**. The number of characters to be returned. Default is 1.

Remarks

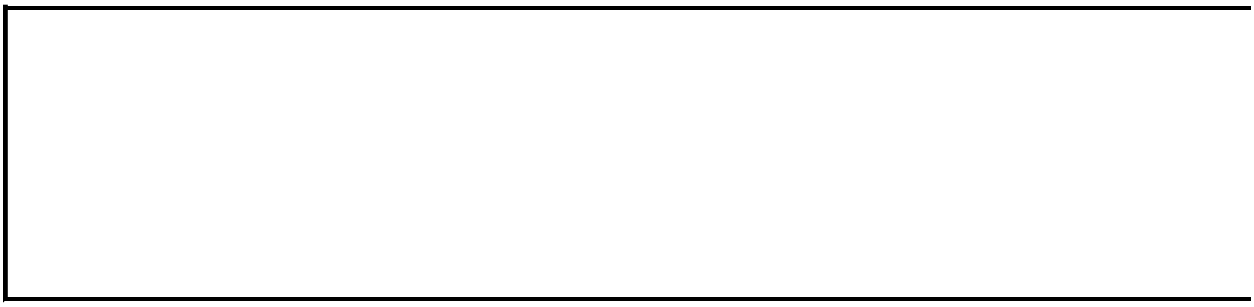
If ***Start*** is greater than the number of characters in the specified text, the returned range starts with the last character in the specified range.

If ***Length*** is greater than the number of characters from the specified starting character to the end of the text, the returned range contains all those characters.

Example

This example sets the text for the first shape on the first page in the active document, and then sets the font of the first two characters to 15 points and bold.

```
Sub CharRange()
    Dim rngCharacters As TextRange
    Set rngCharacters = Application.ActiveDocument.Pages(1).Shapes(1
        .TextFrame.TextRange.InsertBefore(NewText:="Hello World.")
    With rngCharacters.Characters(Start:=1, Length:=2).Font
        .Size = 15
        .Bold = msoTrue
    End With
End Sub
```



Clear Method

DropCap object: Removes the dropped capital letter formatting.

PhoneticGuide object: Removes the phonetic information from Japanese text.

TabStop object: Removes the specified custom tab stop.

FindReplace object: Removes the specified search criteria in a find or replace operation.

expression.**Clear**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example removes the dropped capital letter formatting in the specified text frame.

```
Sub ClearDropCap()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.DropCap.Clear
End Sub
```

ClearAll Method

Clears all the custom tab stops from the specified paragraphs.

expression.**ClearAll**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

To clear an individual tab stop, use the **Clear** method of the **TabStop** object. The **ClearAll** method doesn't clear the default tab stops. To manipulate the default tab stops, use the **DefaultTabStop** property for the document.

Example

This example clears all the custom tab stops in the first shape on the first page of the active publication. This example assumes that the specified shape is a text frame and not another type of shape.

```
Sub ClearAllTabStops()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.ParagraphFormat.Tabs.ClearAll
End Sub
```


Show All

Close Method

 Close method as it applies to the **Document** object.

Closes the current publication and creates a blank one in its place.

expression.**Close**

expression Required. An expression that returns a **Document** object.

Remarks

You can only use the **Close** method on an open **Document** object in another instance of Publisher. Attempting to close the active publication in the current instance of Publisher causes an error.

 Close method as it applies to the **MailMergeDataSource** object.

Closes the specified mail merge data source, cancels the mail merge, and converts all mail merge data fields to plain text.

expression.**Close**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

As it applies to the **Document** object.

This example opens a publication in a new instance of Publisher for modification and then closes the publication. (Note that *Filename* must be replaced with a valid file name for this example to work.)

```
Sub ModifyAnotherPublication()
    ' Create new instance of Publisher.
    Dim appPub As New Publisher.Application

    ' Open publication.
    appPub.Open FileName:="Filename"

    ' Put code here to modify the publication as necessary.

    ' Close the publication.
    appPub.ActiveDocument.Close

    ' Release the other instance of Publisher.
    Set appPub = Nothing
End Sub
```

As it applies to the **MailMergeDataSource** object.

The following example closes the data source for the active mail merge publication.

```
ActiveDocument.MailMerge.DataSource.Close
```


Show All

Collapse Method

Collapses a range or selection to the starting or ending position. After a range or selection is collapsed, the starting and ending points are equal.

expression.**Collapse**(***Direction***)

expression Required. An expression that returns one of the objects in the Applies To list.

Direction Required **PbCollapseDirection**. The direction in which to collapse the range or selection.

PbCollapseDirection can be one of these PbCollapseDirection constants.

pbCollapseEnd

pbCollapseStart

Remarks

If you use **pbCollapseEnd** to collapse a range that refers to an entire paragraph, the range will be located after the ending paragraph mark (the beginning of the next paragraph). However, you can move the range back one character by using the **MoveEnd** method after the range is collapsed.

Example

This example inserts text at the beginning of the second paragraph in the first shape on the first page of the active publication. This example assumes that the specified shape is a text frame and not another type of shape.

```
Sub CollapseRange()
    Dim rngText As TextRange
    Set rngText = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange

    'Collapses range to the end of the range and
    'enters new text and a new paragraph
    With rngText
        .Paragraphs(Start:=1, Length:=1).Collapse Direction:=pbColla
        .Text = "This is a new paragraph." & vbCrLf
    End With
End Sub
```

This example places new text at the end of the first paragraph in the first shape on the first page of the active publication. This example assumes that the specified shape is a text frame and not another type of shape.

```
Sub CollapseSelection()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .Paragraphs(Start:=1, Length:=1).Select

    'Collapses selection to end and moves insertion point back
    'one character, then enters new text
    With Selection.TextRange
        .Collapse Direction:=pbCollapseEnd
        .MoveEnd Unit:=pbTextUnitCharacter, Size:=-1
        .Text = "  This is a new test."
    End With
End Sub
```


Show All

ConvertPublicationType Method

Converts the specified publication to the specified publication type.

expression.**ConvertPublicationType**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **PbPublicationType**. The type of publication to which you want the publication converted.

PbPublicationType can be one of these pbPublicationType constants.

pbTypePrint

pbTypeWeb

Remarks

When a publication is converted, any settings that apply to its previous type remain, but are ignored. For example, converting a print publication to a Web publication results in any advanced print settings being ignored. If the publication is converted back to a print publication, the settings take effect again.

Use the **PublicationType** property of the **Document** object to determine the publication type of a publication.

Example

The following example determines if the active publication is a print publication. If it is, the publication is converted to a Web publication.

```
Sub ChangePublicationType()
    With ActiveDocument
        If .PublicationType = pbTypePrint Then
        .ConvertPublicationType (pbTypeWeb)
        End If
    End With
End Sub
```

ConvertToProcess Method

Converts the specified plate from spot color to process.

expression.**ConvertToProcess**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **ConvertToProcess** method is only accessible if the publication's color mode has been set to process and spot color inks. Returns "Permission Denied" for any other color mode. Use the **ColorMode** property of the **Document** object to specify a publication's color mode.

Returns "Permission Denied" when applied to a process color plate. When the color mode includes process color, the process color inks (black, magenta, yellow and cyan) are the first four plates in the **Plates** collection.

When a plate is converted from spot to process color, all colors in the publication based on the ink that the converted plate represents are converted to process colors.

Example

The following example converts the specified spot color plate to process color. The example assumes the publication's color mode has been specified as spot and process color, and that at least six plates have been defined for the publication.

```
Sub ChangePlateToProcess()

    With ActiveDocument.Plates.Item(6)
        .ConvertToProcess
    End With

End Sub
```

ConvertToShape Method

Creates a shape that has the geometric characteristics of the specified **FreeformBuilder** object. Returns a **Shape** object that represents the new shape.

expression.**ConvertToShape**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

You must apply the **AddNodes** method to a **FreeformBuilder** object at least once before you use the **ConvertToShape** method or an error occurs.

Example

This example adds a freeform with four vertices to the first page in the active publication.

```
' Add a new freeform object.
With ActiveDocument.Shapes _
        .BuildFreeform(EditingType:=msoEditingCorner, _
        X1:=100, Y1:=100)

    ' Add three more nodes and close the polygon.
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingCorner, _
        X1:=200, Y1:=200, X2:=225, Y2:=250, X3:=250, Y3:=200
    .AddNodes SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingAuto, X1:=200, Y1:=100
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=150, Y1:=50
    .AddNodes SegmentType:=msoSegmentLine, _
        EditingType:=msoEditingAuto, X1:=100, Y1:=100

    ' Convert the polygon to a Shape object.
    .ConvertToShape
End With
```

Copy Method

Copies the specified object to the Clipboard.

expression.**Copy**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **Paste** method to paste the contents of the Clipboard.

The **Copy** method can be used on **Shape** objects, but the **Paste** method cannot.

Example

This example copies shapes one and two on page one of the active publication to the Clipboard and then pastes the copies onto page two.

```
With ActiveDocument
    .Pages(1).Shapes.Range(Array(1, 2)).Copy
    .Pages(2).Shapes.Paste
End With
```

This example copies shape one on page one of the active publication to the Clipboard.

```
ActiveDocument.Pages(1).Shapes(1).Copy
```

This example copies the text in shape one on page one of the active publication to the Clipboard.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange.Copy
```

Create Method

Creates a new **PageBackground** object for the specified **Page** object.

expression.**Create**

expression Required. An expression that returns a **PageBackground** object.

Remarks

Use PageBackground.Exists to test if a page already has a background before trying to create a new one. Returns a "Permission denied' error if a background already exists.

Example

The following example tests for the existence of a bckground on the first page of the active document. If a background does not exist then one is created.

```
If ActiveDocument.Pages(1).Background.Exists = False Then
    ActiveDocument.Pages(1).Background.Create
End If
```


Show All

CreatePlateCollection Method

Returns a **Plates** collection that represents a new collection of plates for commercial print separations.

expression.**CreatePlateCollection**(***Mode***)

expression Required. An expression that returns one of the objects in the Applies To list.

Mode Required **PbColorMode**. Indicates the type of plates to create.

PbColorMode can be one of these PbColorMode constants.

pbColorModeBW

pbColorModeDesktop

pbColorModeProcess

pbColorModeSpot

pbColorModeSpotAndProcess

Example

This example creates a new spot-color plate collection and adds a plate to it.

```
Sub AddNewPlates()
    Dim plts As Plates
    Set plts = ActiveDocument.CreatePlateCollection(pbColorModeSpot)
    plts.Add
    With plts(1)
        .Color.RGB = RGB(Red:=255, Green:=0, Blue:=0)
        .Luminance = 4
    End With
End Sub
```

CustomDrop Method

Sets the vertical distance from the edge of the text bounding box to the place where the callout line attaches to the text box.

expression.**CustomDrop**(***Drop***)

expression Required. An expression that returns one of the objects in the Applies To list.

Drop Required **Variant**. The drop distance. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Remarks

The drop distance is normally measured from the top of the text box. However, if the **AutoAttach** property is set to **True** and the text box is to the left of the origin of the callout line (the place to which the callout points), the drop distance is measured from the bottom of the text box.

Example

This example sets the custom drop distance to 14 points, and specifies that the drop distance always be measured from the top. For the example to work, the third shape in the active publication must be a callout.

```
With ActiveDocument.Pages(1).Shapes(3).Callout
    .CustomDrop Drop:=14
    .AutoAttach = False
End With
```

CustomLength Method

Specifies that the first segment of the callout line (the segment attached to the text callout box) retain a fixed length whenever the callout is moved.

expression.**CustomLength**(***Length***)

expression Required. An expression that returns one of the objects in the Applies To list.

Length Required **Variant**. The length of the first segment of the callout. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Remarks

Applying this method sets the **AutoLength** property to **False** and sets the **Length** property to the value specified for the ***Length*** argument.

Use the **AutomaticLength** method to specify that the first segment of the callout line be scaled automatically whenever the callout is moved. Applies only to callouts whose lines consist of more than one segment (types **msoCalloutThree** and **msoCalloutFour**).

Example

This example toggles between an automatically-scaling first segment and one with a fixed length for the callout line for the first shape in the active publication. For the example to work, this shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    If .AutoLength Then
        .CustomLength Length:=50
    Else
        .AutomaticLength
    End If
End With
```

Cut Method

Deletes the specified object and places it on the Clipboard.

expression.**Cut**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **Paste** method to paste the contents of the Clipboard.

The **Copy** method can be used on **Shape** objects, but the **Paste** method cannot.

Example

This example deletes shapes one and two from page one of the active publication, places copies of them on the Clipboard, and then pastes the copies onto page two.

```
With ActiveDocument
    .Pages(1).Shapes.Range(Array(1, 2)).Cut
    .Pages(2).Shapes.Paste
End With
```

This example deletes shape one on page one of the active publication and places a copy of it on the Clipboard.

```
ActiveDocument.Pages(1).Shapes(1).Cut
```

This example deletes the text in shape one on page one of the active publication and places a copy of it on the Clipboard.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange.Cut
```


Show All

Delete Method

 **Delete** method as it applies to the **Plate** object.

Deletes the specified plate.

expression.**Delete**(***PlateReplaceWith, ReplaceTint***)

expression Required. An expression that returns a **Plate** object.

PlateReplaceWith Optional **Plate**. The plate with which to replace the deleted plate.

ReplaceTint Optional **pbReplaceTint**.

ReplaceTint can be one of these **pbReplaceTint** constants.

pbReplaceTintKeepTints Maintain the same tint percentage in the ink represented by the replacement plate as in the deleted plate. For example, replace a 100% tint of yellow with a 100% tint of blue.

pbReplaceTintMaintainLuminosity Maintain the same lightness value in the ink represented by the replacement plate as in the deleted plate. For example, replace a 100% tint of yellow with an approximately 10% tint of blue.

pbReplaceTintUseDefault *Default*

Remarks

Returns "Permission Denied" if you attempt to delete the last plate in the **Plates** collection.

If the **pbReplaceTintMaintainLuminosity** constant is specified, the percentage of replacment ink in each color is calculated based on the luminosity values of the inks represented by the deleted and replacement plates. Publisher performs the following calculation, where *L1* is the deleted ink luminosity, and *L2* is the replacement ink luminosity: (100-*L1*)/(100-*L2*).

For example, red ink has a luminosity of 30, and black has a luminosity of 0. Suppose you replaced the red ink plate in a publication with a black ink plate. If **pbReplaceTintKeepTints** is specified, Publisher performs the following calculation to determine the percentage of black ink for each red color: (100-30)/(100-0). A color that was 100% red would now be 70% black; a color that was 50% red would now be 35% black, and so on.

If the **pbReplaceTintKeepTints** constant is specified, the percentage of the replacement ink in each color is the same as the deleted color. For example, if red ink is replaced with black ink, 100% tint of red is replaced by 100% tint of black, 50% red with 50% black, and so on.

You cannot specify the **pbReplaceTintMaintainLuminosity** or **pbReplaceTintUseDefault** constants if the replacement plate represents an ink that has a higher luminosity (that is, is lighter) than the deleted plate. This is because the lighter ink can not be printed at more than 100%, so it will not be able to match the luminosity of the darker ink.

 **Delete** method as it applies to the **ShapeNodes** object.

Deletes the specified shape node object.

expression.**Delete**(***Index***)

expression Required. An expression that returns a **ShapeNodes** collection.

Index Required **Long**. The number of the shape node to delete.

Delete method as it applies to the **WebHiddenFields** and **WebListBoxItems** objects.

Deletes the specified hidden Web field or Web list box item object.

expression.**Delete**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Long**. The number of the Web field or list box item to delete.

Delete method as it applies to all the other objects in the Applies To list.

Deletes the specified object.

expression.**Delete**

expression Required. An expression that returns one of the other objects in the Applies To list.

Remarks

A run-time error occurs if the specified object does not exist.

Example

 As it applies to the **BorderArtFormat** object.

The following example tests for the existence of BorderArt on each shape for each page of the active publication. If BorderArt exists, it is deleted.

```
Sub DeleteBorderArt()

Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Delete
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

 As it applies to the **Plate** object.

The following example loops through the active publication's plates collection, determines which plates represent inks not used in the publication, and deletes them. This example assumes that at least one of the plates is in use (the Delete method returns "Permission Denied" if you attempt to delete the last plate in the collection.)

```
Sub DeleteUnusedInks()

Dim intCount As Integer

With ActiveDocument.Plates
    For intCount = .Count To 1 Step -1
        With .Item(intCount)
            If .InUse = False Then
                Debug.Print "Name: " & .Name
                .Delete
            End If
        End With
```

```
    Next
End With

End Sub
```

As it applies to the **ShapeNodes** object.

This example deletes the first node in the first shape in the active publication.

```
Sub DeleteNode()
    ActiveDocument.Pages(1).Shapes(1).Nodes.Delete Index:=1
End Sub
```

As it applies to the **Shapes** object.

This example deletes the first shape in the active publication.

```
Sub DeleteShape()
    ActiveDocument.Pages(1).Shapes(1).Delete
End Sub
```

DeleteSetAndInstances Method

Deletes a Web navigation bar set and all instances of it in the current document.

expression.**DeleteSetAndInstances**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Example

The following example iterates through the **WebNavigationBarSets** collection and deletes each set from the active document.

```
Dim objWebNavBarSet As WebNavigationBarSet
For Each objWebNavBarSet In ActiveDocument.WebNavigationBarSets
    objWebNavBarSet.DeleteSetAndInstances
Next objWebNavBarSet
```


Show All

Distribute Method

Evenly distributes the shapes in the specified shape range.

expression.**Distribute**(***DistributeCmd***, ***RelativeTo***)

expression Required. An expression that returns one of the objects in the Applies To list.

DistributeCmd Required **MsoDistributeCmd**. Specifies whether shapes are to be distributed horizontally or vertically.

MsoDistributeCmd can be one of these MsoDistributeCmd constants.
msoDistributeHorizontally
msoDistributeVertically

RelativeTo Required **MsoTriState**. Specifies whether to distribute the shapes evenly over the entire horizontal or vertical space on the page or within the horizontal or vertical space that the range of shapes originally occupies.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this method.
msoFalse Distribute the shapes within the horizontal or vertical space that the range of shapes originally occupies.
msoTriStateMixed Not used with this method.
msoTriStateToggle Not used with this method.
msoTrue Distribute the shapes evenly over the entire horizontal or vertical space on the page.

Remarks

Shapes are distributed so that there is an equal amount of space between one shape and the next. If the shapes are so large that they overlap when distributed over the available space, they are distributed so that there is an equal amount of overlap between one shape and the next.

When ***RelativeTo*** is **msoTrue**, shapes are distributed so that the distance between the two outer shapes and the edges of the page is the same as the distance between one shape and the next. If the shapes must overlap, the two outer shapes are moved to the edges of the page.

When ***RelativeTo*** is **msoFalse**, the two outer shapes are not moved; only the positions of the inner shapes are adjusted.

The z-order of shapes is unaffected by this method.

Example

This example defines a shape range that contains all the AutoShapes on the first page of the active publication and then horizontally distributes the shapes in this range.

```
' Number of shapes on the page.
Dim intShapes As Integer
' Number of AutoShapes on the page.
Dim intAutoShapes As Integer
' An array of the names of the AutoShapes.
Dim arrAutoShapes() As String
' A looping variable.
Dim shpLoop As Shape
' A placeholder variable for the range containing AutoShapes.
Dim shpRange As ShapeRange

With ActiveDocument.Pages(1).Shapes
    ' Count all the shapes on the page.
    intShapes = .Count

    ' Proceed only if there's at least one shape.
    If intShapes > 1 Then
        intAutoShapes = 0
        ReDim arrAutoShapes(1 To intShapes)

        ' Loop through the shapes on the page and add the names
        ' of any AutoShapes to an array.
        For Each shpLoop In ActiveDocument.Pages(1).Shapes
            If shpLoop.Type = msoAutoShape Then
                intAutoShapes = intAutoShapes + 1
                arrAutoShapes(intAutoShapes) = shpLoop.Name
            End If
        Next shpLoop

        ' Proceed only if there's at least one AutoShape.
        If intAutoShapes > 1 Then
            ReDim Preserve arrAutoShapes(1 To intAutoShapes)

            ' Create a shape range containing all the AutoShapes.
            Set shpRange = .Range(Index:=arrAutoShapes)

            ' Distribute the AutoShapes horizontally
            ' in the space they already occupy.
            shpRange.Distribute _
                DistributeCmd:=msoDistributeHorizontally, RelativeTo
```

```
        End If
    End If
End With
```

DoVerb Method

Requests that an OLE object perform one of its verbs.

expression.**DoVerb**(***iVerb***)

expression Required. An expression that returns one of the objects in the Applies To list.

iVerb Required **Long**. The verb to perform.

Remarks

Use the **ObjectVerbs** property to determine the available verbs for an OLE object.

Example

This example performs the first verb for the third shape on the first page of the active publication if the shape is a linked or embedded OLE object.

```
With ActiveDocument.Pages(1).Shapes(3)
    If .Type = pbEmbeddedOLEObject Or _
            .Type = pbLinkedOLEObject Then
        .OLEFormat.DoVerb (1)
    End If
End With
```

This example performs the verb "Open" for the third shape on the first page of the active publication if the shape is an OLE object that supports the verb "Open."

```
Dim strVerb As String
Dim intVerb As Integer

With ActiveDocument.Pages(1).Shapes(3)

    ' Verify that the shape is an OLE object.
    If .Type = pbEmbeddedOLEObject Or _
            .Type = pbLinkedOLEObject Then

        ' Loop through the ObjectVerbs collection
        ' until the "Open" verb is found.
        For Each strVerb In .OLEFormat.ObjectVerbs
            intVerb = intVerb + 1
            If strVerb = "Open" Then

                ' Perform the "Open" verb.
                .OLEFormat.DoVerb iVerb:=intVerb
                Exit For
            End If
        Next strVerb
    End If
End With
```


Show All

Duplicate Method

 **Duplicate** method as it applies to the **Font** object.

Creates a duplicate of the specified **Font** object and then returns the new **Font** object.

expression.**Duplicate**

expression Required. An expression that returns a **Font** object.

 **Duplicate** method as it applies to the **Page** object.

Creates a duplicate of the specified **Page** object and then returns the new **Page** object.

expression.**Duplicate**

expression Required. An expression that returns a **Page** object.

 **Duplicate** method as it applies to the **ParagraphFormat** object.

Creates a duplicate of the specified **ParagraphFormat** object and then returns the new **ParagraphFormat** object.

expression.**Duplicate**

expression Required. An expression that returns a **ParagraphFormat** object.

 **Duplicate** method as it applies to the **Shape** and **ShapeRange** objects.

Creates a duplicate of the specified **Shape** or **ShapeRange** object, adds the new shape or range of shapes to the **Shapes** collection immediately after the shape or range of shapes specified originally, and then returns the new **Shape** or **ShapeRange** object.

expression.**Duplicate**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Example

 As it applies to the **Font** object.

The following example duplicates the character formatting information from the text range in shape one on page one of the active publication and applies it to the text range in shape two.

```
Dim fntTemp As Font

With ActiveDocument.Pages(1)
    Set fntTemp = _
        .Shapes(1).TextFrame.TextRange.Font.Duplicate
    .Shapes(2).TextFrame.TextRange.Font = fntTemp
End With
```

 As it applies to the **Page** object.

The following example duplicates the first page in the publication and then sets properties for the duplicate. A shape is then added to the new page and properties are set for the shape.

```
Dim objPage As Page
Set objPage = ActiveDocument.Pages(1).Duplicate
With objPage
    .Background.Fill.ForeColor.SchemeColor = pbSchemeColorAccent1
    .Shapes.AddShape msoShapeRectangle, 150, 250, 310, 275
    With .Shapes(1)
        .Fill.ForeColor.SchemeColor = pbSchemeColorAccent3
    End With
End With
```

 As it applies to the **ParagraphFormat** object.

The following example duplicates the paragraph formatting information from the text range in shape one on page one of the active publication and applies it to the text range in shape two.

```
Dim pfTemp As ParagraphFormat

With ActiveDocument.Pages(1)
    Set pfTemp = .Shapes(1).TextFrame _
        .TextRange.ParagraphFormat.Duplicate
    .Shapes(2).TextFrame _
        .TextRange.ParagraphFormat = pfTemp
End With
```

 As it applies to the **Shape** and **ShapeRange** objects.

This example adds a new, blank page at the end of the active publication, adds a diamond shape to the new page, duplicates the diamond, and then sets properties for the duplicate. The first diamond will have the default fill color for the active color scheme; the second diamond will be offset from the first one and will have the first accent color for the active color scheme.

```
Dim pgTemp As Page
Dim shpTemp As Shape

Set pgTemp = ActiveDocument.Pages.Add(Count:=1, After:=1)
Set shpTemp = pgTemp.Shapes _
    .AddShape(Type:=msoShapeDiamond, _
    Left:=10, Top:=10, Width:=250, Height:=350)

With shpTemp.Duplicate
    .Left = 150
    .Fill.ForeColor.SchemeColor = pbSchemeColorAccent1
End With
```

EmusToPoints Method

Converts a measurement from emus to points (12700 emus = 1 point). Returns the converted measurement as a **Single**.

expression.**EmusToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The emu value to be converted to points.

Remarks

Use the **PointsToEmus** method to convert measurements in points to emus.

Example

This example converts measurements in emus entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in emus (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " emus = " _
        & Format(Application _
        .EmusToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```

EndConnect Method

Attaches the end of the specified connector to a specified shape.

expression.**EndConnect**(***ConnectedShape***, ***ConnectionSite***)

expression Required. An expression that returns one of the objects in the Applies To list.

ConnectedShape Required **Shape** object. The shape to which Microsoft Publisher attaches the end of the connector. The specified **Shape** object must be in the same **Shapes** collection as the connector.

ConnectionSite Required **Long**. A connection site on the shape specified by ***ConnectedShape***. Must be an integer between 1 and the integer returned by the **ConnectionSiteCount** property of the specified shape. Connection sites are numbered starting from the top of the specified shape and moving counterclockwise around the shape. If you want the connector to automatically find the shortest path between the two shapes it connects, specify any valid integer for this argument and then use the **RerouteConnections** method after the connector is attached to shapes at both ends.

Remarks

If there's already a connection between the end of the connector and another shape, that connection is broken. If the end of the connector isn't already positioned at the specified connecting site, this method moves the end of the connector to the connecting site and adjusts the size and position of the connector.

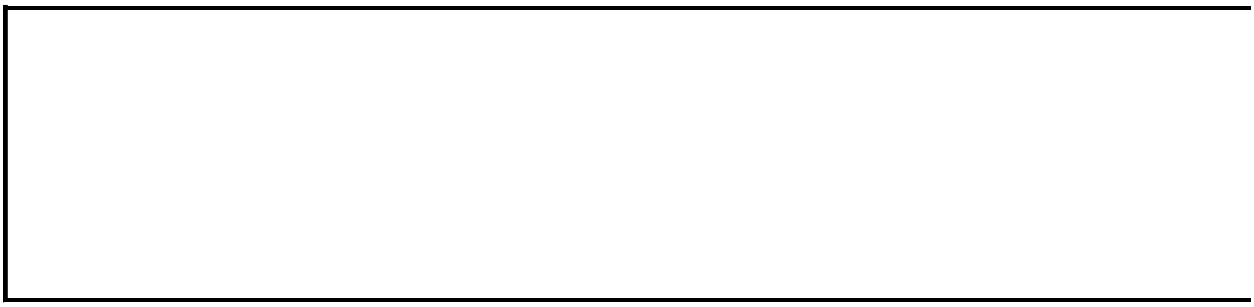
When you attach a connector to an object, the size and position of the connector are automatically adjusted if necessary.

Use the **BeginConnect** method to attach the beginning of the connector to a shape.

Example

This example adds two rectangles to the first page in the active publication and connects them with a curved connector. Note that the **RerouteConnections** method overrides the values you supply for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new curved connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections
    End With

End With
```



EndCustomUndoAction Method

Specifies the end point of a group of actions that are wrapped to create a single undo action. The **BeginCustomUndoAction** method is used to specify the starting point and label (textual description) of the actions used to create the single undo action. The wrapped group of actions can be undone with a single undo.

expression.**EndCustomUndoAction**()

expression Required. An expression that returns a **Document** object.

Remarks

The **BeginCustomUndoAction** method must be called before the **EndCustomUndoAction** method is called. A run-time error is returned if **EndCustomUndoAction** is called before **BeginCustomUndoAction**.

Example

The following example contains two custom undo actions. The first one is created on page four of the active publication. The **BeginCustomUndoAction** method is used to specify the point at which the custom undo action should begin. Six individual actions are performed, and then they are wrapped into one action with the call to **EndCustomUndoAction**.

The text in the text frame that was created within the first custom undo action is then tested to determine whether the font is Verdana. If not, the **Undo** method is called with **UndoActionsAvailable** passed as a parameter. In this case there is only one undo action available. So, the call to **Undo** will only undo one action, but this one action has wrapped six actions into one.

A second undo action is then created, and it could also be undone later with a single undo operation.

This example assumes that the active publication contains at least four pages.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(4)

With theDoc
    ' The following six of actions are wrapped to create one
    ' custom undo action named "Add Rectangle and Courier Text".
    .BeginCustomUndoAction ("Add Rectangle and Courier Text")
    With thePage
        Set theShape = .Shapes.AddShape(msoShapeRectangle, _
            75, 75, 190, 30)
        With theShape.TextFrame.TextRange
             .Font.Size = 14
             .Font.Bold = msoTrue
             .Font.Name = "Courier"
             .Text = "This font is Courier."
        End With
     End With
     .EndCustomUndoAction

    If Not thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Verdan
```

```
        ' This call to Undo will undo all actions that are available
        ' In this case, there is only one action that can be undone.
        .Undo (.UndoActionsAvailable)
        ' A new custom undo action is created with a name of
        ' "Add Balloon and Verdana Text".
        .BeginCustomUndoAction ("Add Balloon and Verdana Text")
        With thePage
            Set theShape = .Shapes.AddShape(msoShapeBalloon, _
                75, 75, 190, 30)
            With theShape.TextFrame.TextRange
                .Font.Size = 11
                .Font.Name = "Verdana"
                .Text = "This font is Verdana."
            End With
        End With
        .EndCustomUndoAction
    End If
End With
```

EndDisconnect Method

Detaches the end of the specified connector from the shape to which it's attached.

expression.**EndDisconnect**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This method doesn't alter the size or position of the connector; the end of the connector remains positioned at a connection site but is no longer connected.

Use the **BeginDisconnect** method to detach the beginning of the connector from a shape.

Example

This example adds two rectangles to the first page in the active publication, attaches them with a connector, automatically reroutes the connector along the shortest path, and then detaches the connector from the rectangles.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=0, EndY:=0) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections

        ' Disconnect the new connector from the rectangles but
        ' leave in place.
        .BeginDisconnect
        .EndDisconnect
    End With

End With
```


Show All

EnterColorMode Method

Accesses the color mode for the publication.

expression.**EnterColorMode**(***Mode***, ***Plates***)

expression Required. An expression that returns one of the objects in the Applies To list.

Mode Required **PbColorMode**. The color mode.

PbColorMode can be one of these PbColorMode constants.
pbColorModeBW
pbColorModeDesktop
pbColorModeProcess
pbColorModeSpot
pbColorModeSpotAndProcess

Plates Optional **Variant**. The plates associated with the color mode. Plates are ignored if the color mode is set to **pbColorModeDesktop**.

Remarks

You can only enter one of the color modes specified by the ***Mode*** argument for each publication. Therefore, if you write a procedure to enter the spot color mode and then write another procedure to enter the black-and-white color mode, only the first procedure executed will run correctly.

Example

This example creates a spot-color plate collection, adds two plates to it, and then enters those plates into the spot color mode.

```
Sub CreateSpotColorMode()
    Dim plArray As Plates

    'Creates a color plate collection,
    'which contains one black plate by default
    Set plArray = ThisDocument.CreatePlateCollection(Mode:=pbColorMo

    'Sets the plate color to red
    plArray(1).Color.RGB = RGB(255, 0, 0)

    'Adds another plate, black by default and
    'sets the plate color to green
    plArray.Add
    plArray(2).Color.RGB = RGB(0, 255, 0)

    'Enters spot-color mode with above
    'two plates in the plates array
    ThisDocument.EnterColorMode Mode:=pbColorModeSpot, Plates:=plArr
End Sub
```


Show All

Execute Method

 As it applies to the **FindReplace** object.

Performs the specified Find or Replace operation.

expression.**Execute**

expression Required. An expression that returns a **FindReplace** object.

Note Be sure to set the **FindText** property before calling the **Execute** method to avoid a run time error.

 As it applies to the **MailMerge** object.

Performs the specified mail merge or catalog merge operation. Returns a **Document** object that represents the new or existing publication specified as the destination of the merge results. Returns **Nothing** if the merge is executed to a printer.

expression.**Execute**(***Pause***, ***Destination***, ***Filename***)

expression Required. An expression that returns a **MailMerge** object.

Pause Required **Boolean**. **True** to have Publisher pause and display a troubleshooting dialog box if a merge error is found. **False** to ignore errors during mail merge or catalog merge.

Destination Optional **PbMailMergeDestination**. The destination of the mail merge or catalog merge results. Specifying **pbSendToPrinter** for a catalog merge results in a run-time error.

PbMailMergeDestination can be one of these PbMailMergeDestination constants.

pbSendToPrinter *Default*

pbMergeToNewPublication

pbMergeToExistingPublication

Filename Optional **String**. The file name of the publication to which you want to append the catalog merge results.

Example

 As it applies to the **FindReplace** object.

This example executes a Find and Replace operation on the active document.

```
Sub ExecuteFindReplace()
    Dim objFindReplace As FindReplace
    Set objFindReplace = ActiveDocument.Find
    With objFindReplace
        .Clear
        .FindText = "library"
        .Execute
    End With
End Sub
```

 As it applies to the **MailMerge** object.

This example executes a mail merge if the active publication is a main document with an attached data source.

```
Sub ExecuteMerge()
    Dim mrgDocument As MailMerge
    Set mrgDocument = ActiveDocument.MailMerge
    If mrgDocument.DataSource.ConnectString <> "" Then
        mrgDocument.Execute Pause:=False
    End If
End Sub
```


Show All

Expand Method

Expands the specified range or selection. Returns or sets a **Long** that represents the number of specified units added to the range or selection.

expression.**Expand**(***Unit***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbTextUnit**. The unit by which to expand the range.

PbTextUnit can be one of these PbTextUnit constants.
pbTextUnitCell
pbTextUnitCharacter
pbTextUnitCharFormat
pbTextUnitCodePoint
pbTextUnitColumn
pbTextUnitLine
pbTextUnitObject
pbTextUnitParaFormat
pbTextUnitParagraph
pbTextUnitRow
pbTextUnitScreen
pbTextUnitSection
pbTextUnitSentence
pbTextUnitStory
pbTextUnitTable
pbTextUnitWindow
pbTextUnitWord

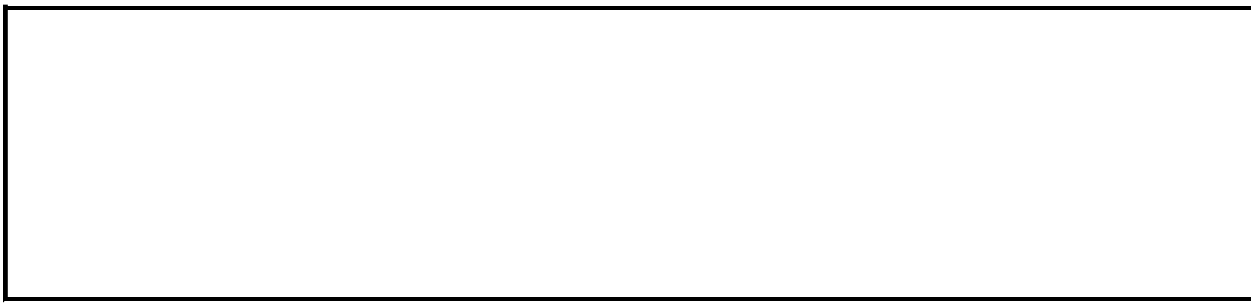
Remarks

The **Expand** method moves both endpoints of a range if necessary; to move only one endpoint of a range, use the **MoveStart** or **MoveEnd** method.

Example

This example creates a range that refers to the first word in the first shape of the active publication, formats the font for the word, and then it expands the range to reference the entire first paragraph and formats the font for the whole line.

```
Sub ExpandRange()
    Dim rngText As TextRange

    Set rngText = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.Words(Start:=1, Length:=1)
    With rngText
        With .Font
            .Size = 20
            .Italic = msoTrue
        End With
        .Expand Unit:=pbTextUnitLine
        .Font.Bold = msoTrue
    End With
End Sub
```



ExportEmailHTML Method

Exports the active page of the publication as an HTML file.

expression.**ExportEmailHTML**(***Filename***)

expression Required. An expression that returns one of the objects in the Applies To list.

Filename Required **String**. The name of the file to which to export the HTML.

Remarks

If the name of an existing HTML file is specified, that file is overwritten.

This method can only be used on the active page of the publication.

Example

The following example sets the first page in the document as the active page, and exports that page to a file. (Note that *PathToFile* must be replaced with a valid file path for this example to work.)

```
Sub ExportEmail()
    Dim strFilePath As String
    strFilePath = "PathToFile"
    With ActiveDocument.ActiveView
        .ActivePage = ActiveDocument.Pages(1)
        .ActivePage.ExportEmailHTML (strFilePath)
    End With

End Sub
```

FindByPageID Method

Returns a **Page** object that represents the page with the specified page ID number. Each page is automatically assigned a unique ID number when it's created. Use the **PageID** property to return a page's ID number.

expression.**FindByPageID**(***PageID***)

expression Required. An expression that returns a **Pages** collection.

PageID Required **Long**. Specifies the ID number of the page you want to return. Publisher assigns this number when the page is created.

Remarks

Unlike the **PageIndex** property, the **PageID** property of a **Page** object won't change when you add pages to or rearrange pages in the publication. Therefore, using the **FindByPageID** method with the page ID number can be a more reliable way to return a specific **Page** object from a **Pages** collection than using the **Item** method with the page's index number.

Example

This example demonstrates how to retrieve the unique ID number for a **Page** object and then use this number to return that **Page** object from the **Pages** collection and add a new shape to the page.

```
Sub FindPage()
    Dim lngPageID As Long

    'Get page ID
    lngPageID = ActiveDocument.Pages.Add(Count:=1, After:=1).PageID

    'Use page ID to add a new shape to the page
    ActiveDocument.Pages.FindByPageID(PageID:=lngPageID) _
        .Shapes.AddShape Type:=msoShape5pointStar, _
        Left:=200, Top:=72, Width:=50, Height:=50

End Sub
```


Show All

FindPlateByInkName Method

 As it applies to the **PrintablePlates** object.

Returns a **PrintablePlate** object that represents the printable plate of the specified ink name.

expression.**FindPlateByInkName**(***InkName***)

expression Required. An expression that returns an **AdvancedPrintOptions** object.

InkName Required **PbInkName**. Specifies the printable plate to return.

PbInkName can be one of these pbInkName constants.
pbInkNameBlack
pbInkNameCyan
pbInkNameMagenta
pbInkNameYellow
pbInkNameSpotColor1
pbInkNameSpotColor2
pbInkNameSpotColor3
pbInkNameSpotColor4
pbInkNameSpotColor5
pbInkNameSpotColor6
pbInkNameSpotColor7
pbInkNameSpotColor8
pbInkNameSpotColor9
pbInkNameSpotColor10
pbInkNameSpotColor11
pbInkNameSpotColor12

Remarks

The **PrintablePlates** collection is generated when a publication's print mode is set to separations. Returns "Permission Denied" when any other print mode is specified.

The **PrintablePlates** collection represents the plates that will actually be printed for the publication, based on:

- The plates (if any) you have defined for the publication.
- The advanced print options specified.

 As it applies to the **Plates** object.

Returns a **Plate** object that represents the plate of the specified ink name.

expression.**FindPlateByInkName**(***InkName***)

expression Required. An expression that returns one of the objects in the Applies To list.

InkName Required **PbInkName**. Specifies the plate to return.

PbInkName can be one of these pbInkName constants.
pbInkNameBlack
pbInkNameCyan
pbInkNameMagenta
pbInkNameYellow
pbInkNameSpotColor1
pbInkNameSpotColor2
pbInkNameSpotColor3
pbInkNameSpotColor4
pbInkNameSpotColor5
pbInkNameSpotColor6
pbInkNameSpotColor7
pbInkNameSpotColor8

pbInkNameSpotColor9
pbInkNameSpotColor10
pbInkNameSpotColor11
pbInkNameSpotColor12

Example

As it applies to the **PrintablePlates** object.

The following example returns a spot color plate and sets several of its properties. The example assumes that separations have been specified as the active publication's print mode.

```
Sub SetPlatePropertiesByInkName()

Dim pplPlate As PrintablePlate
ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

    Set pplPlate = ActiveDocument.AdvancedPrintOptions.PrintablePlat

    With pplPlate
        .Angle = 75
        .Frequency = 133
        .PrintPlate = True
    End With

End Sub
```

As it applies to the **Plates** object.

The following example returns properties for the plate representing the third spot color defined for the active publication.

```
Sub ListPlatePropertiesByInkName()
Dim pplPlate As Plate

    Set pplPlate = ActiveDocument.Plates.FindPlateByInkName(pbInkNam

    With pplPlate
            Debug.Print "Plate Name: " & .Name
            Debug.Print "Index: " & .Index
            Debug.Print "Ink Name: " & .InkName
            Debug.Print "Color: " & .Color
            Debug.Print "Luminance: " & .Luminance
            Debug.Print "In Use?: " & .InUse
    End With
End Sub
```

FindPropertyById Method

Returns a **WizardProperty** object, based on the specified ID, from the collection of wizard properties associated with a publication design or a Design Gallery object's wizard.

expression.**FindPropertyById(*ID*)**

expression Required. An expression that returns one of the objects in the Applies To list.

ID Required **Long**. The ID of the the wizard property to return; corresponds to the **ID** property of the **WizardProperty** object.

Example

The following example changes the settings of the current publication design (Newsletter Wizard) so that the publication has a region dedicated to the customer's address (Customer Address).

```
Sub SetWizardProperties
    Dim wizTemp As Wizard
    Dim wizproTemp As WizardProperty

    Set wizTemp = ActiveDocument.Wizard

    With wizTemp.Properties
        Set wizproTemp = .FindPropertyById(ID:=901)
        wizproTemp.CurrentValueId = 1
    End With

End Sub
```

FindRecord Method

Searches the contents of the specified mail merge data source for text in a particular field. Returns a **Boolean** indicating whether the search text is found; **True** if the search text is found.

expression.**FindRecord**(***FindText***, ***Field***)

expression Required. An expression that returns one of the objects in the Applies To list.

FindText Required **String**. The text to look for.

Field Optional **String**. The name of the field to be searched.

Example

This example displays a merge publication for the first data record in which the FirstName field contains Joe. If the data record is found, the record number is stored in a variable.

```
Sub FindDataSourceRecord()
    Dim dsMain As MailMergeDataSource
    Dim intRecord As Integer

    'Makes the data in the data source records instead of the field
    ActiveDocument.MailMerge.ViewMailMergeFieldCodes = False

    Set dsMain = ActiveDocument.MailMerge.DataSource

    If dsMain.FindRecord(FindText:="Joe", _
            Field:="FirstName") = True Then
        intRecord = dsMain.ActiveRecord
    End If

End Sub
```


Show All

FindShapeByWizardTag Method

Returns a **ShapeRange** object representing one or all of the shapes placed in a publication by a wizard and bearing the specified wizard tag.

expression.**FindShapeByWizardTag**(***WizardTag***, ***Instance***)

expression Required. An expression that returns one of the objects in the Applies To list.

WizardTag Required **PbWizardTag**. Specifies the wizard tag for which to search.

PbWizardTag can be one of these PbWizardTag constants.
pbWizardTagAddress
pbWizardTagAddressGroup
pbWizardTagBriefDescriptionCaption
pbWizardTagBriefDescriptionGraphic
pbWizardTagBriefDescriptionSummary
pbWizardTagBriefDescriptionSummaryPrimary
pbWizardTagBriefDescriptionTitle
pbWizardTagBusinessDescription
pbWizardTagCustomerMailingAddress
pbWizardTagDate
pbWizardTagEAPostalCodeBox
pbWizardTagEAPostalCodeGroup
pbWizardTagEAPostalCodeLine
pbWizardTagFloatingGraphicCaption
pbWizardTagHourTimeDateInformation
pbWizardTagJobTitle
pbWizardTagLinkedStoryPrimary
pbWizardTagLinkedStorySecondary
pbWizardTagLinkedStoryTertiary

pbWizardTagList
pbWizardTagLocation
pbWizardTagLogoGroup
pbWizardTagMainFloatingGraphic
pbWizardTagMainGraphic
pbWizardTagMainTitle
pbWizardTagMapPicture
pbWizardTagMasthead
pbWizardTagNewsletterTitle
pbWizardTagOrganizationName
pbWizardTagOrganizationNameGroup
pbWizardTagPageNumber
pbWizardTagPersonalName
pbWizardTagPersonalNameGroup
pbWizardTagPhoneFaxEmail
pbWizardTagPhoneFaxEmailGroup
pbWizardTagPhoneNumber
pbWizardTagPhotePlaceholderText
pbWizardTagPhotoPlaceholderFrame
pbWizardTagPublicationDate
pbWizardTagQuickPubContent
pbWizardTagQuickPubHeading
pbWizardTagQuickPubMessage
pbWizardTagQuickPubPicture
pbWizardTagReturnAddressLines
pbWizardTagStampBox
pbWizardTagStampBoxOutline
pbWizardTagStory
pbWizardTagStoryCaptionPrimary
pbWizardTagStoryCaptionSecondary
pbWizardTagStoryGraphicPrimary
pbWizardTagStoryGraphicSecondary
pbWizardTagStoryTitle

pbWizardTagTableOfContents
pbWizardTagTableOfContentsTitle
pbWizardTagTagLine
pbWizardTagTagLineGroup
pbWizardTagTime

Instance Optional **Long**. Specifies which instance of a shape with the specified wizard tag is returned. For ***Instance*** equal to *n* , the *n* th instance of a shape with the specified wizard tag is returned. If no value for ***Instance*** is specified, all the shapes with the specified wizard tag are returned.

Example

The following example finds the second instance of a shape with the wizard tag **pbWizardDate** and assigns it to a variable.

```
Dim shpWizardTag As Shape

Set shpWizardTag = ActiveDocument._
    FindShapeByWizardTag(WizardTag:=pbWizardDate, Instance:=2)
```

FindShapesByTag Method

Returns a **ShapeRange** object that represents the shapes with the specified tag.

expression.**FindShapesByTag**(***TagName***)

expression Required. An expression that returns a **Document** object.

TagName Required **String**. The name of the tag.

Example

This example adds two shapes to the first page of the active publication, assigns each a tag, and then enters the name of each tag into the text frame of its assigned shape.

```
Sub FindShape()
    Dim strTag1 As String
    Dim strTag2 As String

    With ActiveDocument.Pages(1).Shapes
        With .AddShape(Type:=msoShape5pointStar, Left:=50, _
                Top:=50, Width:=75, Height:=75)
            strTag1 = .Tags.Add(Name:="Star", _
                Value:="This is a star.").Name
        End With

        With .AddShape(Type:=msoShapeHeart, Left:=100, _
                Top:=100, Width:=75, Height:=75)
            strTag2 = .Tags.Add(Name:="Heart", _
                Value:="This is a heart.").Name
        End With
    End With

    With ActiveDocument
        .FindShapesByTag(TagName:=strTag1).TextFrame _
            .TextRange.Text = strTag1
        .FindShapesByTag(TagName:=strTag2).TextFrame _
            .TextRange.Text = strTag2
    End With
End Sub
```


Show All

Flip Method

Flips the specified shape around its horizontal or vertical axis, or flips all the shapes in the specified shape range around their horizontal or vertical axes.

expression.**Flip**(***FlipCmd***)

expression Required. An expression that returns one of the objects in the Applies To list.

FlipCmd Required **MsoFlipCmd**. Specifies whether the shape is flipped horizontally or vertically.

MsoFlipCmd can be one of these MsoFlipCmd constants.

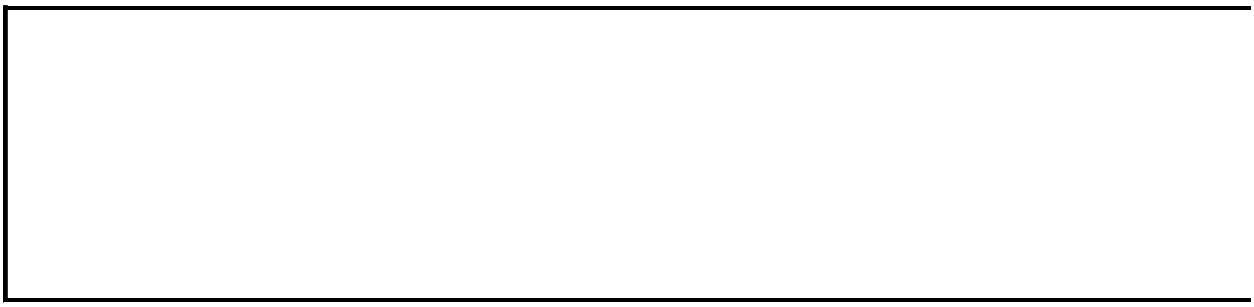
msoFlipHorizontal

msoFlipVertical

Example

This example adds a triangle to the first page of the active publication, duplicates the triangle, and then flips the duplicate triangle vertically and makes it red.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeRightTriangle, _
        Left:=10, Top:=10, Width:=50, Height:=50) _
        .Duplicate
    .Fill.ForeColor.RGB = RGB(255, 0, 0)
    .Flip msoFlipVertical
End With
```




Show All

GetHeight Method

Returns the height of the shape or shape range as a **Single** in the specified units.

expression.**GetHeight**(***Unit***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbUnitType**. The units in which to return the height.

PbUnitType can be one of these PbUnitType constants.
pbUnitCM
pbUnitEmu
pbUnitFeet
pbUnitHa
pbUnitInch
pbUnitKyu
pbUnitMeter
pbUnitPica
pbUnitPoint
pbUnitTwip

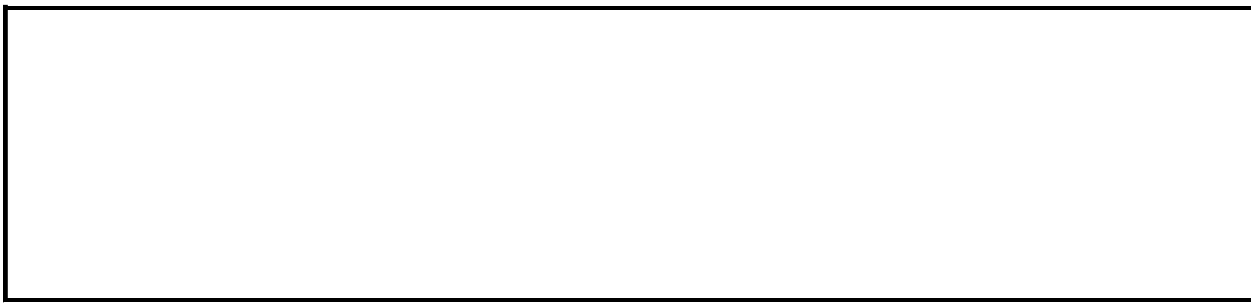
Remarks

Use the **GetWidth** method to return the width of a shape or shape range.

Example

The following example displays the height and width in inches (to the nearest hundredth) of the shape range consisting of all the shapes on the first page of the active publication.

```
With ActiveDocument.Pages(1).Shapes.Range
    MsgBox "Height of all shapes: " _
        & Format(.GetHeight(Unit:=pbUnitInch), "0.00") _
        & " in" & vbCr _
        & "Width of all shapes: " _
        & Format(.GetWidth(Unit:=pbUnitInch), "0.00") _
        & " in"
End With
```




Show All

GetLeft Method

Returns the distance of the shape’s or shape range's left edge from the left edge of the leftmost page in the current view as a **Single** in the specified units.

expression.**GetLeft**(***Unit***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbUnitType**. The units in which to return the distance.

PbUnitType can be one of these PbUnitType constants.
pbUnitCM
pbUnitEmu
pbUnitFeet
pbUnitHa
pbUnitInch
pbUnitKyu
pbUnitMeter
pbUnitPica
pbUnitPoint
pbUnitTwip

Remarks

Use the **GetTop** method to return the distance of a shape's or shape range's top edge from the top edge of the leftmost page in the current view.

Example

The following example displays the distances from the left and top edges of the leftmost page to the left and top edges of shape range consisting of all the shapes on the first page. The distances are expressed in inches (to the nearest hundredth).

```
With ActiveDocument.Pages(1).Shapes.Range
    MsgBox "Distance from left: " _
        & Format(.GetLeft(Unit:=pbUnitInch), "0.00") _
        & " in" & vbCr _
        & "Distance from top: " _
        & Format(.GetTop(Unit:=pbUnitInch), "0.00") _
        & " in"
End With
```


Show All

GetScriptName Method

Returns a **String** that represents the name of the font script being used in a text range.

expression.**GetScriptName**(***Script***)

expression Required. An expression that returns one of the objects in the Applies To list.

Script Required **PbFontScriptType**. The script name.

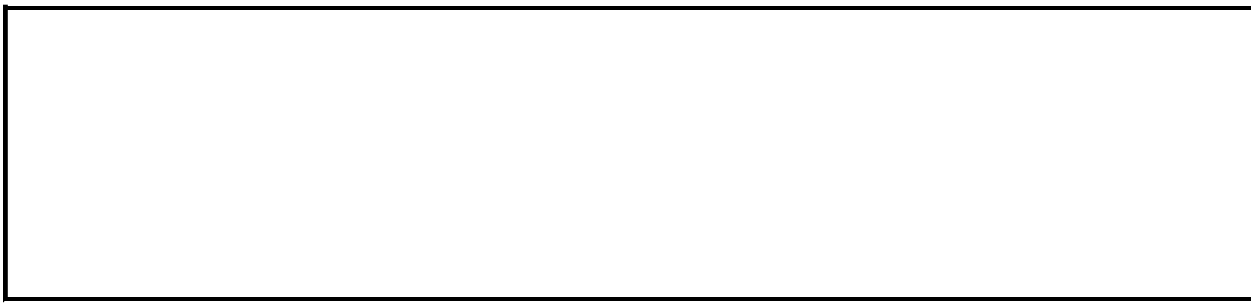
PbFontScriptType can be one of these PbFontScriptType constants.
pbFontScriptArabic
pbFontScriptArmenian
pbFontScriptAsciiLatin
pbFontScriptAsciiSym
pbFontScriptBengali
pbFontScriptBopomofo
pbFontScriptBraille
pbFontScriptCanadianAbor
pbFontScriptCherokee
pbFontScriptCurrency
pbFontScriptCyrillic
pbFontScriptDefault
pbFontScriptDevanagari
pbFontScriptEthiopic
pbFontScriptEUDC
pbFontScriptGeorgian
pbFontScriptGreek
pbFontScriptGujarati
pbFontScriptGurmukhi
pbFontScriptHalfWidthKana

pbFontScriptHan
pbFontScriptHangul
pbFontScriptHanSurrogate
pbFontScriptHebrew
pbFontScriptKana
pbFontScriptKannada
pbFontScriptKhmer
pbFontScriptLao
pbFontScriptLatin
pbFontScriptMalayalam
pbFontScriptMixed
pbFontScriptMongolian
pbFontScriptMyanmar
pbFontScriptNonHanSurrogate
pbFontScriptOgham
pbFontScriptOriya
pbFontScriptRunic
pbFontScriptSinhala
pbFontScriptSyriac
pbFontScriptTamil
pbFontScriptTelugu
pbFontScriptThaana
pbFontScriptThai
pbFontScriptTibetan
pbFontScriptYi

Example

This example verifies that the default font script in use for the specified text range is Tahoma and, if not, sets it as the default font script.

```
Sub GetScript()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.Font
        If .GetScriptName(Script:=pbFontScriptDefault) <> "Tahoma" T
            .SetScriptName Script:=pbFontScriptDefault, _
                FontName:="Tahoma"
        End If
    End With
End Sub
```




Show All

GetTop Method

Returns the distance of the shape’s or shape range's top edge from the top edge of the leftmost page in the current view as a **Single** in the specified units.

expression.**GetTop**(***Unit***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbUnitType**. The units in which to return the distance.

PbUnitType can be one of these PbUnitType constants.
pbUnitCM
pbUnitEmu
pbUnitFeet
pbUnitHa
pbUnitInch
pbUnitKyu
pbUnitMeter
pbUnitPica
pbUnitPoint
pbUnitTwip

Remarks

Use the **GetLeft** method to return the distance of a shape's or shape range's left edge from the left edge of the leftmost page in the current view.

Example

The following example displays the distances from the left and top edges of the leftmost page to the left and top edges of shape range consisting of all the shapes on the first page. The distances are expressed in inches (to the nearest hundredth).

```
With ActiveDocument.Pages(1).Shapes.Range
    MsgBox "Distance from left: " _
        & Format(.GetLeft(Unit:=pbUnitInch), "0.00") _
        & " in" & vbCr _
        & "Distance from top: " _
        & Format(.GetTop(Unit:=pbUnitInch), "0.00") _
        & " in"
End With
```


Show All

GetWidth Method

Returns the width of the shape or shape range as a **Single** in the specified units.

expression.**GetWidth**(***Unit***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbUnitType**. The units in which to return the width.

PbUnitType can be one of these PbUnitType constants.
pbUnitCM
pbUnitEmu
pbUnitFeet
pbUnitHa
pbUnitInch
pbUnitKyu
pbUnitMeter
pbUnitPica
pbUnitPoint
pbUnitTwip

Remarks

Use the **GetHeight** method to return the width of a shape or shape range.

Example

The following example displays the height and width in inches (to the nearest hundredth) of the shape range consisting of all the shapes on the first page of the active publication.

```
With ActiveDocument.Pages(1).Shapes.Range
    MsgBox "Height of all shapes: " _
        & Format(.GetHeight(Unit:=pbUnitInch), "0.00") _
        & " in" & vbCr _
        & "Width of all shapes: " _
        & Format(.GetWidth(Unit:=pbUnitInch), "0.00") _
        & " in"
End With
```

Group Method

Groups the shapes in the specified shape range. Returns the grouped shapes as a single **Shape** object.

expression.**Group**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

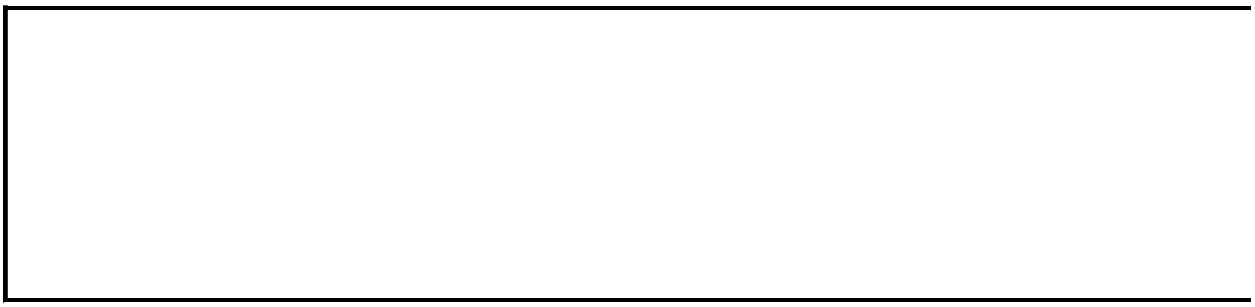
The specified range must contain more than one shape, or an error occurs.

Because a group of shapes is treated as a single shape, grouping and ungrouping shapes changes the number of items in the **Shapes** collection and changes the index numbers of items that come after the affected items in the collection.

Example

This example adds two shapes to the first page of the active publication, groups the two new shapes, sets the fill for the group, rotates the group, and sends the group to the back of the drawing layer.

```
With ActiveDocument.Pages(1).Shapes

    ' Add two shapes to the page.
    .AddShape(Type:=msoShapeCan, _
        Left:=50, Top:=10, Width:=100, Height:=200).Name = "shpOne"
    .AddShape(Type:=msoShapeCube, _
        Left:=150, Top:=250, Width:=100, Height:=200).Name = "shpTwo

    ' Group the shapes and change the formatting for the whole group
    With .Range(Index:=Array("shpOne", "shpTwo")).Group
        .Fill.PresetTextured PresetTexture:=msoTextureBlueTissuePape
        .Rotation = 45
        .ZOrder ZOrderCmd:=msoSendToBack
    End With

End With
```



Grow Method

Increases the font size to the next available size.

expression.**Grow**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the selection or range contains more than one font size, each size is increased to the next available setting.

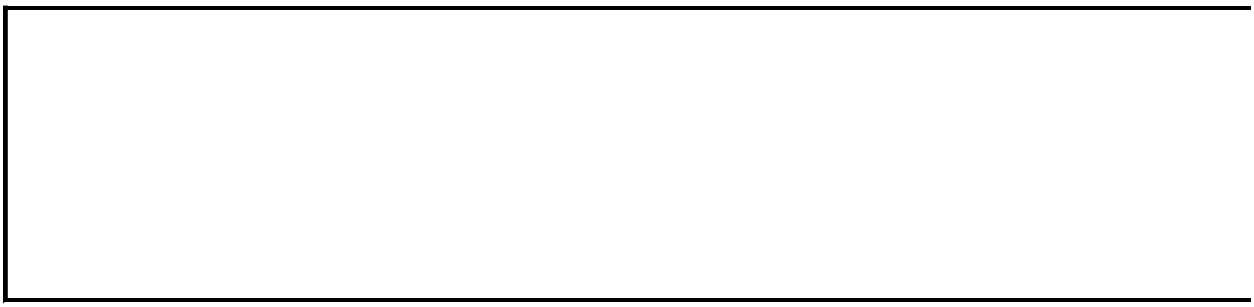
Example

This example increases the font size of the fourth word in a new textbox.

```
Sub GrowFont()
    Dim shpText As Shape
    Dim intResponse As Integer

    Set shpText = ActiveDocument.Pages(1).Shapes.AddTextbox( _
        Orientation:=pbTextOrientationHorizontal, Left:=100, _
        Top:=100, Width:=200, Height:=100)

    With shpText.TextFrame.TextRange
        .Text = "This is a test of the Grow method."
        Do Until intResponse = vbNo
            intResponse = MsgBox("Do you want to increase the " & _
                "size of the font?", vbYesNo)
            If intResponse = vbYes Then
                .Words(4).Font.Grow
            End If
        Loop
    End With
End Sub
```

This example increases the font size of the selected text.

```
Sub IncreaseFontSizeOfSelectedText()
    If Selection.Type = pbSelectionText Then
        Selection.TextRange.Font.Grow
    Else
        MsgBox "You need to select some text."
    End If
End Sub
```




Show All

Help Method

Displays online Help information.

expression.**Help**(***HelpType***)

expression Required. An expression that returns one of the objects in the Applies To list.

HelpType Required **PbHelpType**. The type of help to display.

PbHelpType can be one of these PbHelpType constants.

pbHelp Displays the **Help Topics** dialog box.

pbHelpActiveWindow Displays Help describing the command associated with the active view or pane.

pbHelpPSSHelp Displays product support information.

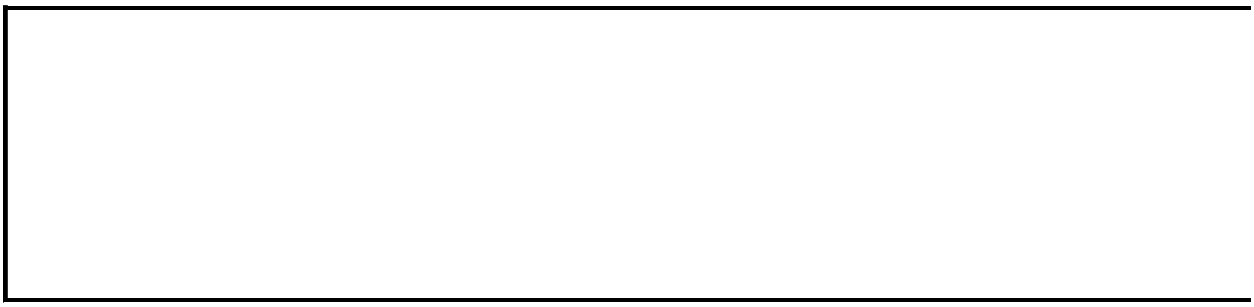
Remarks

Some of the constants listed above may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

Example

This example displays a list of topics for troubleshooting printing problems.

```
Sub ShowPrintTroubleshooter()
    Application.Help (HelpType:=pbHelpPrintTroubleshooter)
End Sub
```



InchesToPoints Method

Converts a measurement from inches to points (1 inch = 72 points). Returns the converted measurement as a **Single**.

expression.**InchesToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The inches value to be converted to points.

Remarks

Use the **PointsToInches** method to convert measurements in points to inches.

Example

This example converts measurements in inches entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in inches (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " in = " _
        & Format(Application _
        .InchesToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```

IncrementBrightness Method

Changes the brightness of the picture by the specified amount.

expression.**IncrementBrightness**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Single**. Specifies how much to change the value of the **Brightness** property for the picture. A positive value makes the picture brighter; a negative value makes the picture darker. Valid values are between – 1 and 1.

Remarks

You cannot adjust the brightness of a picture past the upper or lower limit for the **Brightness** property. For example, if the **Brightness** property is initially set to 0.9 and you specify 0.3 for the ***Increment*** argument, the resulting brightness level will be 1.0, which is the upper limit for the **Brightness** property, instead of 1.2.

Use the **Brightness** property to set the absolute brightness of the picture.

Example

This example creates a duplicate of the first shape in the active publication and then moves and darkens the duplicate. For the example to work, the shape must be either a picture or an OLE object representing a picture.

```
With ActiveDocument.Pages(1).Shapes(1).Duplicate
    .PictureFormat.IncrementBrightness Increment:=-0.2
    .IncrementLeft Increment:=50
    .IncrementTop Increment:=50
End With
```

IncrementContrast Method

Changes the contrast of the picture by the specified amount.

expression.**IncrementContrast**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Single**. Specifies how much to change the value of the **Contrast** property for the picture. A positive value increases the contrast; a negative value decreases the contrast. Valid values are between – 1 and 1.

Remarks

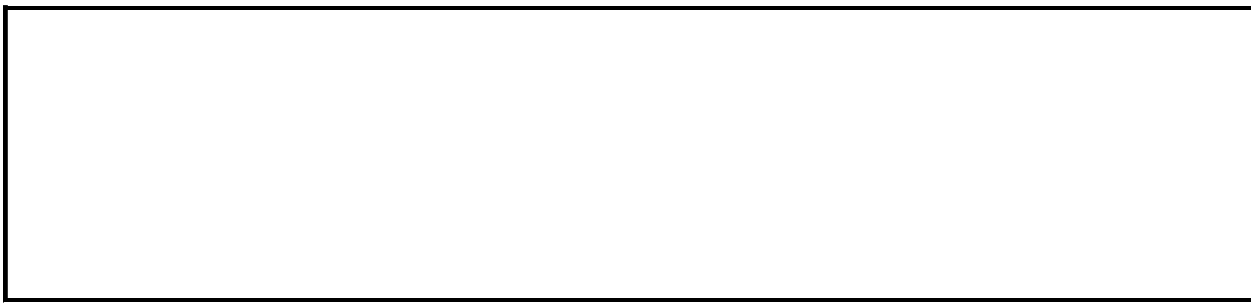
You cannot adjust the contrast of a picture past the upper or lower limit for the **Contrast** property. For example, if the **Contrast** property is initially set to 0.9 and you specify 0.3 for the ***Increment*** argument, the resulting contrast level will be 1.0, which is the upper limit for the **Contrast** property, instead of 1.2.

Use the **Contrast** property to set the absolute contrast for the picture.

Example

This example increases the contrast for all pictures on the first page of the active publication that aren't already set to maximum contrast.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    If shpLoop.Type = msoPicture Then
        shpLoop.PictureFormat.IncrementContrast Increment:=0.1
    End If
Next shpLoop
```



IncrementLeft Method

Moves the specified shape or shape range horizontally by the specified distance.

expression.**IncrementLeft**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Variant**. The horizontal distance to move the shape or shape range. A positive value moves the shape or shape range to the right; a negative value moves it to the left. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

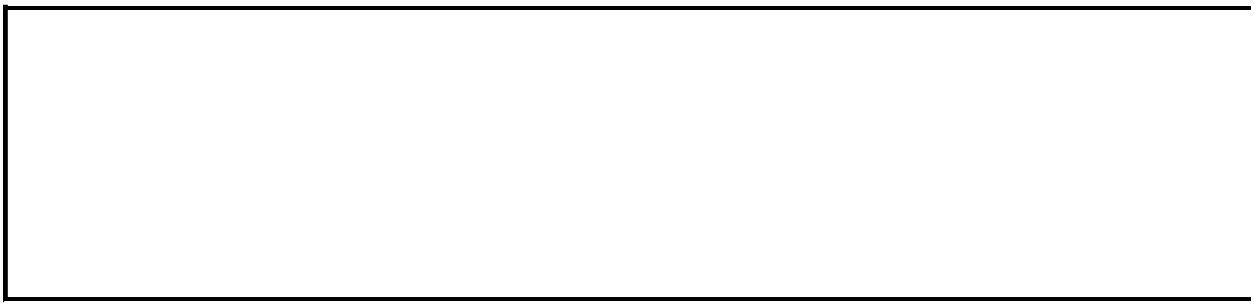
Remarks

Use the **IncrementTop** method to move shapes or shape ranges vertically.

Example

This example duplicates the first shape on the active publication, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
With ActiveDocument.Pages(1).Shapes(1).Duplicate
    .Fill.PresetTextured PresetTexture:=msoTextureGranite
    .IncrementLeft Increment:=70
    .IncrementTop Increment:=-50
    .IncrementRotation Increment:=30
End With
```



IncrementOffsetX Method

Incrementally changes the horizontal offset of the shadow by the specified distance.

expression.**IncrementOffsetX**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Variant**. Specifies how far the shadow offset is to be moved horizontally. A positive value moves the shadow to the right; a negative value moves it to the left. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Remarks

Use the **OffsetX** property to set the absolute horizontal shadow offset.

Use the **IncrementOffsetY** method to change a shadow's vertical offset.

Example

This example moves the shadow for the third shape in the active publication to the left by 3 points.

```
ActiveDocument.Pages(1).Shapes(3).Shadow _
    .IncrementOffsetX Increment:=-3
```

IncrementOffsetY Method

Incrementally changes the vertical offset of the shadow by the specified distance.

expression.**IncrementOffsetY**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Variant**. Specifies how far the shadow offset is to be moved vertically. A positive value moves the shadow down; a negative value moves it up. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Remarks

Use the **OffsetY** property to set the absolute vertical shadow offset.

Use the **IncrementOffsetX** method to change a shadow's horizontal offset.

Example

This example moves the shadow for the third shape in the active publication up by 3 points.

```
ActiveDocument.Pages(1).Shapes(3).Shadow _
    .IncrementOffsetY Increment:=-3
```

IncrementRotation Method

Changes the rotation of the specified shape around the z-axis (extends outward from the plane of the publication) by the specified number of degrees.

expression.**IncrementRotation**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Single**. Specifies how far the shape is to be rotated around the z-axis, in degrees. A positive value rotates the shape clockwise; a negative value rotates it counterclockwise. Valid values are between – 360 and 360.

Remarks

Use the **Rotation** property to set the absolute rotation of the shape.

To rotate a three-dimensional shape around the x-axis (horizontal) or the y-axis (vertical), use the **IncrementRotationX** method or the **IncrementRotationY** method, respectively.

Example

This example duplicates the first shape on the active publication, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
With ActiveDocument.Pages(1).Shapes(1).Duplicate
    .Fill.PresetTextured PresetTexture:=msoTextureGranite
    .IncrementLeft Increment:=70
    .IncrementTop Increment:=-50
    .IncrementRotation Increment:=30
End With
```

IncrementRotationX Method

Changes the rotation of the specified shape around the x-axis (horizontal) by the specified number of degrees.

expression.**IncrementRotationX**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Single**. Specifies by how many degrees to rotate the shape around the x-axis. Can be a value from – 90 through 90. A positive value tilts the shape up; a negative value tilts it down.

Remarks

Use the **RotationX** property to set the absolute rotation of the shape around the x-axis.

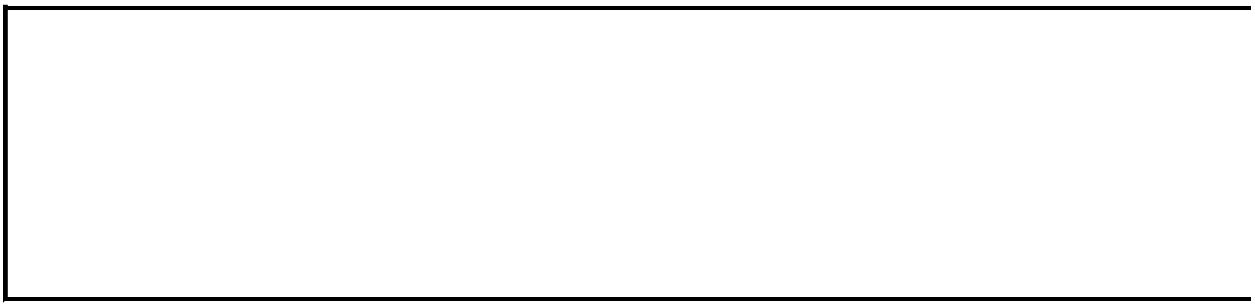
You cannot adjust the rotation around the x-axis of the specified shape past the upper or lower limit for the **RotationX** property (90 degrees to – 90 degrees). For example, if the **RotationX** property is initially set to 80 and you specify 40 for the ***Increment*** argument, the resulting rotation will be 90 (the upper limit for the **RotationX** property) instead of 120.

To change the rotation of a shape around the y-axis (vertical), use the **IncrementRotationY** method. To change the rotation around the z-axis (extends outward from the plane of the publication), use the **IncrementRotation** method.

Example

This example tilts the first shape in the active publication up 10 degrees. The shape must be an extruded shape for you to see the effect of this code.

```
ActiveDocument.Pages(1).Shapes(1).ThreeD _
    .IncrementRotationX Increment:=10
```



IncrementRotationY Method

Changes the rotation of the specified shape around the y-axis (vertical) by the specified number of degrees.

expression.**IncrementRotationY**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Single**. Specifies by how many degrees to rotate the shape around the y-axis. Can be a value from – 90 through 90. A positive value tilts the shape to the left; a negative value tilts it to the right.

Remarks

Use the **RotationY** property to set the absolute rotation of the shape around the y-axis.

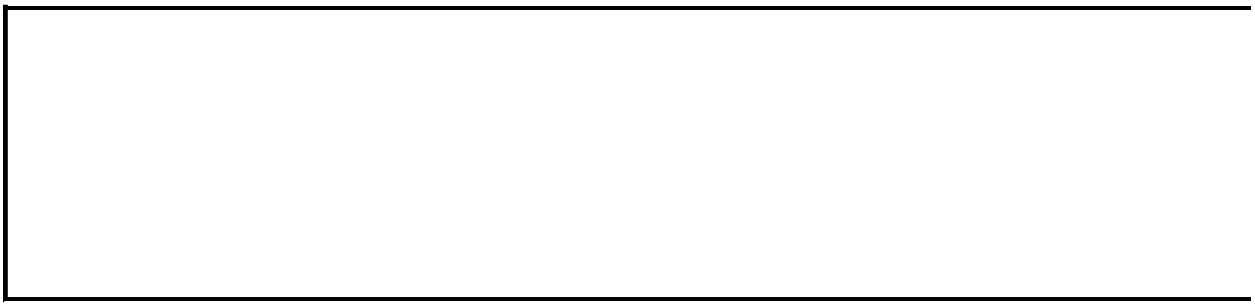
You cannot adjust the rotation around the y-axis of the specified shape past the upper or lower limit for the **RotationY** property (90 degrees to – 90 degrees). For example, if the **RotationY** property is initially set to 80 and you specify 40 for the ***Increment*** argument, the resulting rotation will be 90 (the upper limit for the **RotationY** property) instead of 120.

To change the rotation of a shape around the x-axis (horizontal), use the **IncrementRotationX** method. To change the rotation around the z-axis (extends outward from the plane of the publication), use the **IncrementRotation** method.

Example

This example tilts the first shape in the active publication 10 degrees to the right. The shape must be an extruded shape for you to see the effect of this code.

```
ActiveDocument.Pages(1).Shapes(1).ThreeD _
    .IncrementRotationY Increment:=-10
```



IncrementTop Method

Moves the specified shape or shape range vertically by the specified distance.

expression.**IncrementTop**(***Increment***)

expression Required. An expression that returns one of the objects in the Applies To list.

Increment Required **Variant**. The vertical distance to move the shape or shape range. A positive value moves the shape or shape range down; a negative value moves it up. Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Remarks

Use the **IncrementLeft** method to move shapes or shape ranges horizontally.

Example

This example duplicates the first shape on the active publication, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
With ActiveDocument.Pages(1).Shapes(1).Duplicate
    .Fill.PresetTextured PresetTexture:=msoTextureGranite
    .IncrementLeft Increment:=70
    .IncrementTop Increment:=-50
    .IncrementRotation Increment:=30
End With
```


Show All

Insert Method

 As it applies to the **MailMergeDataField** object .

Adds a **Shape** object that represents a picture data field inserted into the publication's catalog merge area.

expression.**Insert**(***Range***)

expression Required. An expression that returns a **MailMergeDataField** object.

Range Optional **TextRange**.

Remarks

Returns "Permission Denied" for text data fields. Before a data field is inserted into a publication's catalog merge area using the **Insert** method, the field must be defined as a picture data field. Use the **FieldType** property of the **MailMergeDataField** object to specify a data field's type.

Use the **InsertMailMergeField** method of the **TextRange** object to add a text data field to a text box in the publication's catalog merge area.

This method corresponds to inserting picture fields into the catalog merge area in **Step 3: Create your catalog merge template** of the **Mail and Catalog Merge Wizard**.

 As it applies to the **ShapeNodes** collection.

Inserts a new segment after the specified node of the freeform drawing.

expression.**Insert**(***Index***, ***SegmentType***, ***EditingType***, ***X1***, ***Y1***, ***X2***, ***Y2***, ***X3***, ***Y3***)

expression Required. An expression that returns one of the above objects.

Index Required **Long**. The number of the node after which the new node is to be inserted.

SegmentType Required **MsoSegmentType**. The type of segment to be added.

MsoSegmentType can be one of these MsoSegmentType constants.
msoSegmentCurve
msoSegmentLine

EditingType Required **MsoEditingType**. Specifies the editing type of the new node.

MsoEditingType can be one of these MsoEditingType constants.
msoEditingAuto Adds a node type appropriate to the segments being connected.

msoEditingCorner Adds a corner node.
msoEditingSmooth Not used with this method.
msoEditingSymmetric Not used with this method.

X1 Required **Variant**. If the ***EditingType*** of the new segment is **msoEditingAuto**, this argument specifies the horizontal distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new node is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the first control point for the new segment.

Y1 Required **Variant**. If the ***EditingType*** of the new segment is **msoEditingAuto**, this argument specifies the vertical distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new node is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the first control point for the new segment.

X2 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the second control point for the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, do not specify a value for this argument.

Y2 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the second control point for the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, do not specify a value for this argument.

X3 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of the new segment is **msoEditingAuto**, do not specify a value for this argument.

Y3 Optional **Variant**. If the ***EditingType*** of the new segment is **msoEditingCorner**, this argument specifies the vertical distance from the upper-left corner of the page to the end point of the new segment. If the ***EditingType*** of

the new segment is **msoEditingAuto**, do not specify a value for this argument.

Remarks

For the ***X1***, ***Y1***, ***X2***, ***Y2***, ***X3***, and ***Y3*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

Example

 As it applies to the **MailMergeDataField** object.

This example defines a data field as a picture data field, inserts it into the catalog merge area of the specified publication, and sizes and positions the picture data field. This example assumes the publication has been connected to a data source, and a catalog merge area has been added to the publication.

```
Dim pbPictureField1 As Shape

    'Define the field as a picture data type
    With ThisDocument.MailMerge.DataSource.DataFields
        .Item("Photo:").FieldType = pbMailMergeDataFieldPicture
    End With

    'Insert a picture field, then size and position it
    Set pbPictureField1 = ThisDocument.MailMerge.DataSource.DataFiel
        With pbPictureField1
            .Height = 100
            .Width = 100
            .Top = 85
            .Left = 375
        End With
```

 As it applies to the **ShapeNodes** collection.

This example adds a smooth node with a curved segment after node four in the third shape in the active publication. The shape must be a freeform drawing with at least four nodes.

```
With ActiveDocument.Pages(1).Shapes(3).Nodes
    .Insert Index:=4, _
        SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingAuto, _
        X1:=210, Y1:=100
End With
```

InsertAfter Method

Returns a **TextRange** object that represents text appended to the end of a text range.

expression.**InsertAfter**(***NewText***)

expression Required. An expression that returns one of the objects in the Applies To list.

NewText Required **String**. The text to be inserted.

Example

This example adds Microsoft Publisher's build number to the end of the first shape on the first page of the active publication. This example assumes the specified shape is a text frame and not another type of shape.

```
Sub AppendText()
    With ActiveDocument.Pages(1).Shapes(1)
        .TextFrame.TextRange.InsertAfter _
            NewText:="Microsoft Publisher Build : " & Build
    End With
End Sub
```

InsertBefore Method

Returns a **TextRange** object that represents text appended to the beginning of a text range.

expression.**InsertBefore**(***NewText***)

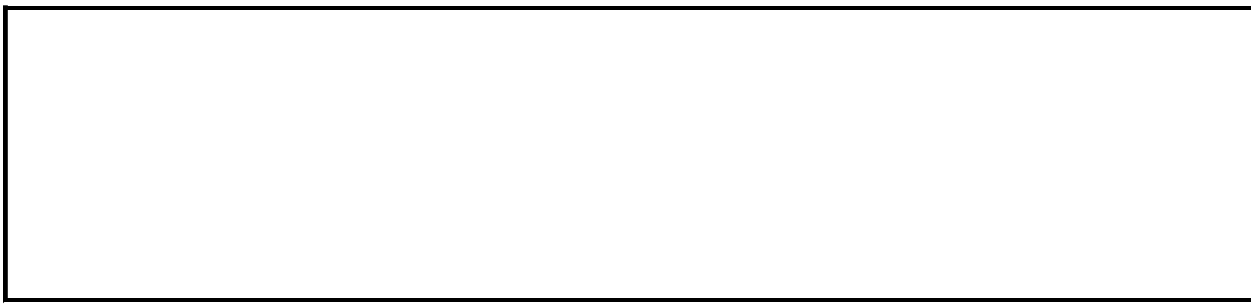
expression Required. An expression that returns one of the objects in the Applies To list.

NewText Required **String**. The text to be inserted.

Example

This example adds Microsoft Publisher's build number and a paragraph break to the beginning of first shape on the first page of the active publication. This example assumes the specified shape is a text frame and not another type of shape.

```
Sub InsertTextBefore()
    With ActiveDocument.Pages(1).Shapes(1)
        .TextFrame.TextRange.InsertBefore _
            NewText:="Microsoft Publisher Build : " & Build & vbCrLf
    End With
End Sub
```




Show All

InsertDateTime Method

Returns a **TextRange** object that represents the date and time inserted into a specified text range.

expression.**InsertDateTime**(***Format***, ***InsertAsField***, ***InsertAsFullWidth***, ***Language***, ***Calendar***)

expression Required. An expression that returns a **TextRange** object.

Format Required **PbDateTimeFormat**. A format for the date and time.

PbDateTimeFormat can be one of these PbDateTimeFormat constants.

pbDateEnglish
pbDateISO
pbDateLong
pbDateLongDay
pbDateMon_Yr
pbDateMonthYr
pbDateShort
pbDateShortAbb
pbDateShortAlt
pbDateShortMon
pbDateShortSlash
pbDateTimeEastAsia1
pbDateTimeEastAsia2
pbDateTimeEastAsia3
pbDateTimeEastAsia4
pbDateTimeEastAsia5
pbTime24
pbTimeDatePM
pbTimeDateSecPM
pbTimePM

pbTimeSec24
pbTimeSecPM

InsertAsField Optional **Boolean**. **True** for Microsoft Publisher to update date and time whenever opening the publication. Default is **False**.

InsertAsFullWidth Optional **Boolean**. **True** to insert the specified information as double-byte digits. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed. Default is **False**.

Language Optional **MsoLanguageID**. The language in which to display the date or time.

MsoLanguageID can be one of these MsoLanguageID constants.

msoLanguageIDAfrikaans
msoLanguageIDAlbanian
msoLanguageIDAmharic
msoLanguageIDArabic
msoLanguageIDArabicAlgeria
msoLanguageIDArabicBahrain
msoLanguageIDArabicEgypt
msoLanguageIDArabicIraq
msoLanguageIDArabicJordan
msoLanguageIDArabicKuwait
msoLanguageIDArabicLebanon
msoLanguageIDArabicLibya
msoLanguageIDArabicMorocco
msoLanguageIDArabicOman
msoLanguageIDArabicQatar
msoLanguageIDArabicSyria
msoLanguageIDArabicTunisia
msoLanguageIDArabicUAE
msoLanguageIDArabicYemen
msoLanguageIDArmenian

msoLanguageIDAssamese
msoLanguageIDAzeriCyrillic
msoLanguageIDAzeriLatin
msoLanguageIDBasque
msoLanguageIDBelgianDutch
msoLanguageIDBelgianFrench
msoLanguageIDBengali
msoLanguageIDBrazilianPortuguese
msoLanguageIDBulgarian
msoLanguageIDBurmese
msoLanguageIDByelorussian
msoLanguageIDCatalan
msoLanguageIDCherokee
msoLanguageIDChineseHongKong
msoLanguageIDChineseMacao
msoLanguageIDChineseSingapore
msoLanguageIDCroatian
msoLanguageIDCzech
msoLanguageIDDanish
msoLanguageIDDivehi
msoLanguageIDDutch
msoLanguageIDEdo
msoLanguageIDEnglishAUS
msoLanguageIDEnglishBelize
msoLanguageIDEnglishCanadian
msoLanguageIDEnglishCaribbean
msoLanguageIDEnglishIreland
msoLanguageIDEnglishJamaica
msoLanguageIDEnglishNewZealand
msoLanguageIDEnglishPhilippines
msoLanguageIDEnglishSouthAfrica
msoLanguageIDEnglishTrinidad
msoLanguageIDEnglishUK

msoLanguageIDEnglishUS
msoLanguageIDEnglishZimbabwe
msoLanguageIDEstonian
msoLanguageIDFaeroese
msoLanguageIDFarsi
msoLanguageIDFilipino
msoLanguageIDFinnish
msoLanguageIDFrench
msoLanguageIDFrenchCameroon
msoLanguageIDFrenchCanadian
msoLanguageIDFrenchCotedIvoire
msoLanguageIDFrenchLuxembourg
msoLanguageIDFrenchMali
msoLanguageIDFrenchMonaco
msoLanguageIDFrenchReunion
msoLanguageIDFrenchSenegal
msoLanguageIDFrenchWestIndies
msoLanguageIDFrenchZaire
msoLanguageIDFrisianNetherlands
msoLanguageIDFulfulde
msoLanguageIDGaelicIreland
msoLanguageIDGaelicScotland
msoLanguageIDGalician
msoLanguageIDGeorgian
msoLanguageIDGerman
msoLanguageIDGermanAustria
msoLanguageIDGermanLiechtenstein
msoLanguageIDGermanLuxembourg
msoLanguageIDGreek
msoLanguageIDGuarani
msoLanguageIDGujarati
msoLanguageIDHausa
msoLanguageIDHawaiian

msoLanguageIDHebrew
msoLanguageIDHindi
msoLanguageIDHungarian
msoLanguageIDIbibio
msoLanguageIDIcelandic
msoLanguageIDIgbo
msoLanguageIDIndonesian
msoLanguageIDInuktitut
msoLanguageIDItalian
msoLanguageIDJapanese
msoLanguageIDKannada
msoLanguageIDKanuri
msoLanguageIDKashmiri
msoLanguageIDKazakh
msoLanguageIDKhmer
msoLanguageIDKirghiz
msoLanguageIDKonkani
msoLanguageIDKorean
msoLanguageIDKyrgyz
msoLanguageIDLao
msoLanguageIDLatin
msoLanguageIDLatvian
msoLanguageIDLithuanian
msoLanguageIDMacedonian
msoLanguageIDMalayalam
msoLanguageIDMalayBruneiDarussalam
msoLanguageIDMalaysian
msoLanguageIDMaltese
msoLanguageIDManipuri
msoLanguageIDMarathi
msoLanguageIDMexicanSpanish
msoLanguageIDMixed
msoLanguageIDMongolian

msoLanguageIDNepali
msoLanguageIDNone *default*
msoLanguageIDNoProofing
msoLanguageIDNorwegianBokmol
msoLanguageIDNorwegianNynorsk
msoLanguageIDOriya
msoLanguageIDOromo
msoLanguageIDPashto
msoLanguageIDPolish
msoLanguageIDPortuguese
msoLanguageIDPunjabi
msoLanguageIDRhaetoRomanic
msoLanguageIDRomanian
msoLanguageIDRomanianMoldova
msoLanguageIDRussian
msoLanguageIDRussianMoldova
msoLanguageIDSamiLappish
msoLanguageIDSanskrit
msoLanguageIDSerbianCyrillic
msoLanguageIDSerbianLatin
msoLanguageIDSesotho
msoLanguageIDSimplifiedChinese
msoLanguageIDSindhi
msoLanguageIDSindhiPakistan
msoLanguageIDSinhalese
msoLanguageIDSlovak
msoLanguageIDSlovenian
msoLanguageIDSomali
msoLanguageIDSorbian
msoLanguageIDSpanish
msoLanguageIDSpanishArgentina
msoLanguageIDSpanishBolivia
msoLanguageIDSpanishChile

msoLanguageIDSpanishColombia
msoLanguageIDSpanishCostaRica
msoLanguageIDSpanishDominicanRepublic
msoLanguageIDSpanishEcuador
msoLanguageIDSpanishElSalvador
msoLanguageIDSpanishGuatemala
msoLanguageIDSpanishHonduras
msoLanguageIDSpanishModernSort
msoLanguageIDSpanishNicaragua
msoLanguageIDSpanishPanama
msoLanguageIDSpanishParaguay
msoLanguageIDSpanishPeru
msoLanguageIDSpanishPuertoRico
msoLanguageIDSpanishUruguay
msoLanguageIDSpanishVenezuela
msoLanguageIDSutu
msoLanguageIDSwahili
msoLanguageIDSwedish
msoLanguageIDSwedishFinland
msoLanguageIDSwissFrench
msoLanguageIDSwissGerman
msoLanguageIDSwissItalian
msoLanguageIDSyriac
msoLanguageIDTajik
msoLanguageIDTamazight
msoLanguageIDTamazightLatin
msoLanguageIDTamil
msoLanguageIDTatar
msoLanguageIDTelugu
msoLanguageIDThai
msoLanguageIDTibetan
msoLanguageIDTigrignaEritrea
msoLanguageIDTigrignaEthiopic

msoLanguageIDTraditionalChinese
msoLanguageIDTsonga
msoLanguageIDTswana
msoLanguageIDTurkish
msoLanguageIDTurkmen
msoLanguageIDUkrainian
msoLanguageIDUrdu
msoLanguageIDUzbekCyrillic
msoLanguageIDUzbekLatin
msoLanguageIDVenda
msoLanguageIDVietnamese
msoLanguageIDWelsh
msoLanguageIDXhosa
msoLanguageIDYi
msoLanguageIDYiddish
msoLanguageIDYoruba
msoLanguageIDZulu

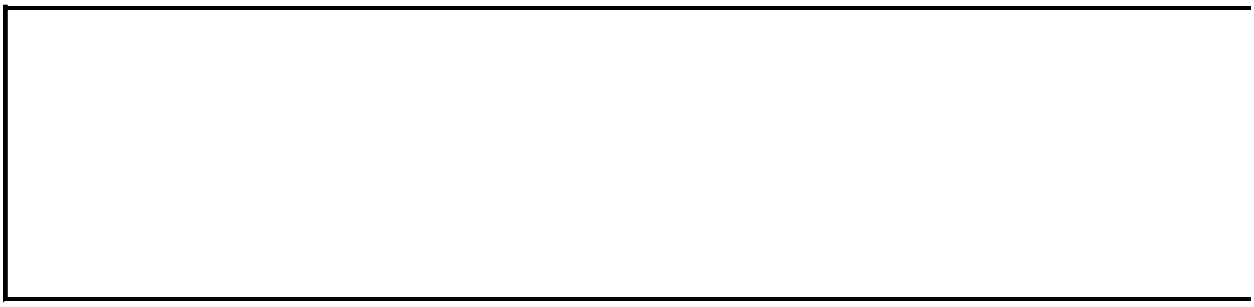
Calendar Optional **PbCalendarType**. The calendar type to use when displaying the date or time.

PbCalendarType can be one of these PbCalendarType constants.
pbCalendarTypeArabicHijri
pbCalendarTypeChineseNational
pbCalendarTypeHebrewLunar
pbCalendarTypeJapaneseEmperor
pbCalendarTypeKoreanDanki
pbCalendarTypeSakaEra
pbCalendarTypeThaiBuddhist
pbCalendarTypeTranslitEnglish
pbCalendarTypeTranslitFrench
pbCalendarTypeWestern *default*

Example

This example inserts a field for the current date at the cursor position.

```
Sub InsertDateField()
    Selection.TextRange.InsertDateTime Format:=pbDateLong, _
        InsertAsField:=True
End Sub
```




Show All

InsertMailMergeField Method

Returns a **TextRange** object that represents a text data field for a mail merge or catalog merge.

expression.**InsertMailMergeField**(***varIndex***)

expression Required. An expression that returns one of the objects in the Applies To list.

varIndex Required **Variant**. The name or index of the data field in the datasource.

Remarks

For a publication's catalog merge area to contain text data fields, it must first contain at least one text box to contain the text data fields.

Example

This example inserts a **LastName** field at the cursor position. This example assumes that the active publication is a mail merge publication and that the insertion point is somewhere inside a text box.

```
Sub InsertMergeField()
    Selection.TextRange.InsertMailMergeField varIndex:="LastName"
End Sub
```

This example adds a text box to the specified publication's catalog merge area, and then inserts a text data field into the text box. This example assumes that the specified publication is connected to a data source, and that it contains a catalog merge area.

```
Set pbTextBox1 = ThisDocument.Pages(1).Shapes.AddTextbox(1, 100, 100
pbTextBox1.AddToCatalogMergeArea

With pbTextBox1.TextFrame.TextRange
        .Text = "List Price: "
        .InsertMailMergeField "List Price"
End With
```


Show All

InsertPageNumber Method

Returns a **TextRange** object that represents a page number field in a publication.

expression.**InsertPageNumber**(***Type***)

expression Required. An expression that returns one of the objects in the Applies To list.

Type Optional **PbPageNumberType**. Specifies whether the page number is the current page number or the next or previous page number of a linked text box.

PbPageNumberType can be one of these PbPageNumberType constants.

pbPageNumberCurrent *default*

pbPageNumberNextInStory Inserts the page number of the next linked text box.

pbPageNumberPreviousInStory Inserts the page number of the previous linked text box.

Example

This example inserts a page number field in a shape on the master page so that the current page number appears at the top of each page.

```
Sub PageNumberShape()
    With ActiveDocument.MasterPages(1).Shapes _
            .AddShape(Type:=msoShape5pointStar, Left:=36, _
            Top:=36, Width:=50, Height:=50)
        With .TextFrame.TextRange
            .InsertPageNumber
            .ParagraphFormat.Alignment = pbParagraphAlignmentCenter
        End With
        .Fill.ForeColor.RGB = RGB(Red:=125, Green:=125, Blue:=255)
    End With
End Sub
```

InsertSymbol Method

Returns a **TextRange** object that represents a symbol inserted in place of the specified range or selection.

expression.**InsertSymbol**(***FontName***, ***CharIndex***)

expression Required. An expression that returns one of the objects in the Applies To list.

FontName Required **String**. The name of the font that contains the symbol.

CharIndex Required **Long**. The Unicode character number for the specified symbol.

Remarks

If you don't want to replace the range or selection, use the **Collapse** method before you use this method.

Example

This example inserts a double-headed arrow at the insertion point.

```
Sub InsertArrow()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .Paragraphs(Start:=1, Length:=1).Select

    With Selection.TextRange
        .Collapse Direction:=pbCollapseStart
        .InsertSymbol FontName:="Symbol", CharIndex:=171
    End With

End Sub
```

IsValidObject Method

Determines whether the specified object variable references a valid object and returns a **Boolean** value: **True** if the specified variable that references an object is valid, **False** if the object referenced by the variable has been deleted.

expression.**IsValidObject**(***Object***)

expression Required. An expression that returns one of the objects in the Applies To list.

Object Required **Object**. A variable that references an object.

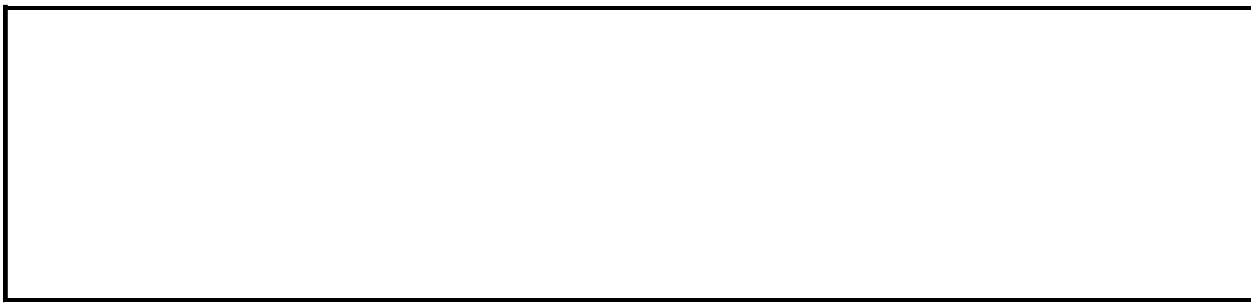
Example

This example formats the line of a valid object.

```
Sub ValidShape(shpObject As Shape)

    If Application.IsValidObject(Object:=shpObject) = True Then
        With shpObject.Line
            .DashStyle = msoLineRoundDot
            .ForeColor.RGB = RGB(Red:=158, Green:=50, Blue:=208)
            .Weight = 5
        End With
    End If

End Sub
```

Use the following subroutine to call the above subroutine.

```
Sub CallValidShape()
    Call ValidShape(shpObject:=ActiveDocument.Pages(1).Shapes(2))
End Sub
```




Show All

Item Method

 **Item** method as it applies to the **InlineShapes** collection.

Returns a **Shape** object that represents an inline shape contained in a text range. This method is the default member of the **InlineShapes** collection.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Variant**. The index position or name of the object to return. If ***Index*** is an integer, the index into the collection is 1-based. If ***Index*** is a string, the name of the shape is used as the index. An automation error is returned if the index or name does not represent a shape in the collection.

 Item method as it applies to the **MailMergeDataFields** object.

Returns a **MailMergeDataField** object from the specified **MailMergeDataFields** object.

expression.**Item**(***varIndex***)

expression Required. An expression that returns one of the above objects.

varIndex Required **Variant**. The number or name of the field to return.

 Item method as it applies to the **CellRange**, **Columns**, **Fields**, **MailMergeFilters**, **ObjectVerbs**, **Rows**, **Stories**, and **TabStops** objects.

Returns an individual object in a specified collection.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Long**. The number of the object to return.

 Item method as it applies to the **WebHiddenFields** and **WebListBoxItems** objects.

Returns a **String** corresponding to the value of a hidden field in a Web form or a list item in a Web list box control.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Variant**. The number or name of the field or list box item to return.

 **Item** method as it applies to all the other objects in the Applies To list.

Returns an individual object in a specified collection.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Variant**. The number or name of the field or list box item to return.

Example

 As it applies to the **InlineShapes** collection.

This example finds the first inline shape in a text range and flips it vertically.

```
Dim theShape As Shape

Set theShape = ActiveDocument.Pages(1).Shapes(1)

With theShape.TextFrame.Story.TextRange
    With .InlineShapes.Item(1)
        .Flip (msoFlipVertical)
    End With
End With
```

 As it applies to the **CellRange** object.

This example returns the first cell from a **CellRange** object.

```
Dim cllTemp As Cell

Set cllTemp = ActiveDocument.Pages(Index:=1) _
    .Shapes(1).Table.Cells.Item(Index:=1)
```

 As it applies to the **Columns** object.

This example returns the first column from a **Columns** object.

```
Dim colTemp As Column

Set colTemp = ActiveDocument.Pages(Index:=1) _
    .Shapes(1).Table.Columns.Item(Index:=1)
```

 As it applies to the **Fields** object.

This example returns the first field from a **Fields** object.

```
Dim fldTemp As Field
```

```
Set fldTemp = ActiveDocument.Pages(Index:=1) _
    .Shapes(1).TextFrame.TextRange.Fields.Item(Index:=1)
```

 As it applies to the **GroupShapes**, **ShapeRange**, and **Shapes** object.

This example returns the first shape inside a grouped shape.

```
Dim shpTemp As Shape

Set shpTemp = ActiveDocument.Pages(Index:=1) _
    .Shapes(1).GroupItems.Item(Index:=1)
```

 As it applies to the **MailMergeMappedDataFields** object.

This example returns the "City" field from a mapped data fields object.

```
Dim mmfTemp As MailMergeMappedDataField

Set mmfTemp = ActiveDocument.MailMerge _
    .DataSource.MappedDataFields.Item(Index:="City")
```

 As it applies to the **TextStyles** object.

This example returns the "Normal" text style from the active publication.

```
Dim txtStyle As TextStyle

Set txtStyle = ActiveDocument.TextStyles.Item(Index:="Normal")
```

LaunchWebService Method

Launches the Microsoft Office eServices Portal.

expression.**LaunchWebService**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example launches the Microsoft Office eServices Portal.

```
Application.LaunchWebService
```

Lines Method

Returns a **TextRange** object that represents the specified lines.

expression.**Lines**(***Start***, ***Length***)

expression Required. An expression that returns one of the objects in the Applies To list.

Start Required **Long**. The first line in the returned range.

Length Optional **Long**. The number of lines to be returned. Default is 1.

Remarks

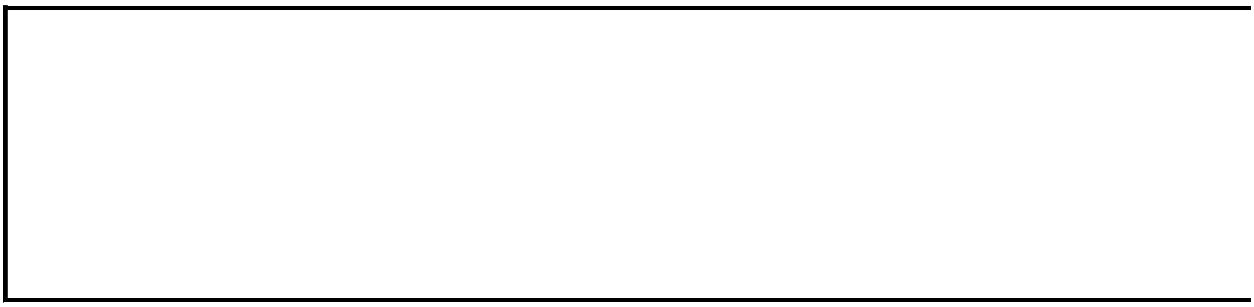
If ***Start*** is greater than the number of lines in the specified text, the returned range starts with the last line in the specified range.

If ***Length*** is greater than the number of lines from the specified starting line to the end of the text, the returned range contains all those lines.

Example

This example replaces the first three lines of the first shape on the first page with the specified string.

```
Sub ReplaceLines()
    Dim rngText As TextRange
    Set rngText = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.Lines(Start:=1, Length:=3)

    rngText.Text = "This is replacement text." & vbCrLf

End Sub
```



LinesToPoints Method

Converts a measurement from lines to points (1 line = 12 points). Returns the converted measurement as a **Single**.

expression.**LinesToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The line value to be converted to points.

Remarks

This method assumes a measurement in 12-point lines— the actual size of any text in the publication has no effect on the conversion factor.

Use the **PointsToLines** method to convert measurements in points to lines.

Example

This example converts measurements in lines to measurements in points, demonstrating that the font size in the current selection has no bearing on the conversion factor. Some text must be selected in the active publication for this example to work.

```
Dim strOutput As String

' Set text size to 10 points.
Selection.TextRange.Font.Size = 10

' Display result for one line of text.
strOutput = "1 line = " _
    & Format(Application _
    .LinesToPoints(Value:=1), _
    "0.00") & " points"
```


Show All

Merge Method

 Merge method as it applies to the **Cell** object.

Merges the specified table cell with another cell. The result is a single table cell.

expression.**Merge**(***MergeTo***)

expression Required. An expression that returns a **Cell** object.

MergeTo Required **Cell** object. The cell to be merged with; must be adjacent to the specified cell or an error occurs.

 Merge method as it applies to the **CellRange** object.

Merges the specified table cells with one another. The result is a single table cell.

expression.**Merge**

expression Required. An expression that returns a **CellRange** object; must be a rectangular region of cells or an error occurs.

Example

 As it applies to the **Cell** object.

This example merges the first two cells of the first column of the specified table.

```
Sub MergeCell()
    With ActiveDocument.Pages(1).Shapes(2).Table
        .Rows(1).Cells(1).Merge MergeTo:=.Rows(2).Cells(1)
    End With
End Sub
```

 As it applies to the **CellRange** object.

This example merges the first two cells in the first two rows of the specified table.

```
Sub MergeCells()
    ActiveDocument.Pages(1).Shapes(2).Table _
        .Cells(StartRow:=1, StartColumn:=1, _
        EndRow:=2, EndColumn:=2).Merge
End Sub
```

MillimetersToPoints Method

Converts a measurement from millimeters to points (1 mm = 2.835 points). Returns the converted measurement as a **Single**.

expression.**MillimetersToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The millimeter value to be converted to points.

Remarks

Use the **PointsToMillimeters** method to convert measurements in points to millimeters.

Example

This example converts measurements in millimeters entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in millimeters (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " mm = " _
        & Format(Application _
        .Mill imetersToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```


Show All

Move Method

 **Move** method as it applies to the **Page** object.

Moves the specified page to the specified index in the **Pages** collection.

expression.**Move**(***Page***, [***After***])

expression Required. An expression that returns a **Page** object.

Page Required **Long**. The index number of the **Pages** collection where the specified page will be moved.

After Optional **Boolean**. **True** if the page will be inserted after the specified index number of the Pages collection specified by the Page parameter. Deafult is **True**.

 **Move** method as it applies to the **TextRange** object.

Collapses the specified range to its start position or end position and then moves the collapsed object by the specified number of units. This method returns a **Long** that represents the number of units by which the object was actually moved, or it returns 0 (zero) if the move was unsuccessful.

expression.**Move**(***Unit***, ***Size***)

expression Required. An expression that returns one of the above objects.

Unit Required **PbTextUnit**. The unit by which the collapsed range or selection is to be moved.

PbTextUnit can be one of these PbTextUnit constants.

pbTextUnitCell

pbTextUnitCharacter

pbTextUnitCharFormat

pbTextUnitCodePoint

pbTextUnitColumn
pbTextUnitLine
pbTextUnitObject
pbTextUnitParaFormat
pbTextUnitParagraph
pbTextUnitRow
pbTextUnitScreen
pbTextUnitSection
pbTextUnitSentence
pbTextUnitStory
pbTextUnitTable
pbTextUnitWindow
pbTextUnitWord

Size Required **Long**. The number of units by which the specified range or selection is to be moved. If ***Size*** is a positive number, the object is collapsed to its end position and moved forward in the document by the specified number of units. If ***Size*** is a negative number, the object is collapsed to its start position and moved backward by the specified number of units. You can also control the collapse direction by using the **Collapse** method before using the **Move** method.

 **Move** method as it applies to the **Window** object.

Moves the active document window.

expression.**Move**(***Left***, ***Top***)

expression Required. An expression that returns one of the above objects.

Left Required **Long**. The horizontal screen position of the specified window.

Top Required **Long**. The vertical screen position of the specified window.

Remarks

If the application window is either maximized or minimized, this method will return an error.

Example

 As it applies to the **Page** object.

This example moves the first page of the publication before the third page of the publication. This example assumes that there are at least three pages in the document.

```
ActiveDocument.Pages(1).Move page:=3, After:=False
```

 As it applies to the **TextRange** object.

This example collapses the specified range and inserts a new sentence at the beginning of the range.

```
Sub MoveText()
    Dim rngText As TextRange
    Set rngText = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Words(Start:=1, Length:=5)
    With rngText
        .Move Unit:=pbTextUnitParagraph, Size:=-1
        .Text = "This adds new text to the beginning of the range.
    End With
End Sub
```

 As it applies to the **Window** object.

This example checks the state of the application window, and if it is neither maximized nor minimized, moves the window to the upper left corner of the screen.

```
Sub MoveWindow()
    With ActiveWindow
        If .WindowState = pbWindowStateNormal Then
            .Move Left:=50, Top:=50
        End If
    End With
End Sub
```


Show All

MoveEnd Method

Moves the ending character position of a range. This method returns a **Long** that represents the number of units the range or selection actually moved or returns 0 (zero) if the move was unsuccessful.

expression.**MoveEnd**(***Unit***, ***Size***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbTextUnit**. The unit by which the collapsed range or selection is to be moved.

PbTextUnit can be one of these PbTextUnit constants.
pbTextUnitCell
pbTextUnitCharacter
pbTextUnitCharFormat
pbTextUnitCodePoint
pbTextUnitColumn
pbTextUnitLine
pbTextUnitObject
pbTextUnitParaFormat
pbTextUnitParagraph
pbTextUnitRow
pbTextUnitScreen
pbTextUnitSection
pbTextUnitSentence
pbTextUnitStory
pbTextUnitTable
pbTextUnitWindow
pbTextUnitWord

Size Required **Long**. The number of units to move. If this number is positive,

the ending character position is moved forward in the document. If this number is negative, the end is moved backward. If the ending position overtakes the starting position, the range collapses and both character positions move together.

Remarks

Use the **MoveStart** method to move the starting character position for a range.

Example

This example sets a text range, moves the range's starting and ending character positions, and then formats the font for the range.

```
Sub MoveStartEnd()
    Dim rngText As TextRange

    Set rngText = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Paragraphs(Start:=3, Length:=1)

    With rngText
        .MoveStart Unit:=pbTextUnitLine, Size:=-2
        .MoveEnd Unit:=pbTextUnitLine, Size:=1
        With .Font
            .Bold = msoTrue
            .Size = 15
        End With
    End With

End Sub
```

MoveIntoTextFlow Method

Moves a given shape into the text flow defined by **TextRange**. The shape will always be inserted inline at the beginning of the text flow.

expression.**MoveIntoTextFlow**(***Range***)

expression Required. An expression that returns one of the objects in the Applies To list.

Range Required **TextRange**. The range of text before which the given shape is inserted.

Remarks

The **MoveIntoTextFlow** method will fail if the shape to be moved is already inline or if it is not a valid inline shape type. Invalid inline shape types include:

- Inline shapes
- Grouped shapes
- HTML fragments
- Smart objects
- Chained text boxes

Example

The following example checks if the second shape on the second page of the publication is inline, and if it is not, inserts it inline at the beginning of the text flow of the given text range.

```
Dim theShape As Shape
Dim theRange As TextRange

Set theRange = ActiveDocument.Pages(2).Shapes(1).TextFrame.TextRange
Set theShape = ActiveDocument.Pages(2).Shapes(2)

If Not theShape.IsInline = msoTrue Then
    theShape.MoveIntoTextFlow Range:=theRange
End If
```

MoveOutOfTextFlow Method

Moves a given inline shape out of its containing text range, defined by **TextRange**, and makes the shape fixed.

expression.**MoveOutOfTextFlow**()

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

An automation error is returned if the shape to be moved is not already inline.

After the **MoveOutOfTextFlow** method is called on an inline shape, the shape will maintain its position on the page, but it will no longer be inline.

Example

The following example moves the first inline shape contained in a given text range out of the text flow.

```
Dim theShape As Shape

Set theShape = ActiveDocument.Pages(2).Shapes(1) _
    .TextFrame.TextRange.InlineShapes(1)

theShape.MoveOutOfTextFlow
```


Show All

MoveStart Method

Moves the start position of the specified range. This method returns a **Long** that indicates the number of units by which the start position or the range or selection actually moved, or it returns 0 (zero) if the move was unsuccessful.

expression.**MoveStart**(***Unit***, ***Size***)

expression Required. An expression that returns one of the objects in the Applies To list.

Unit Required **PbTextUnit**. The unit by which the collapsed range or selection is to be moved.

PbTextUnit can be one of these PbTextUnit constants.
pbTextUnitCell
pbTextUnitCharacter
pbTextUnitCharFormat
pbTextUnitCodePoint
pbTextUnitColumn
pbTextUnitLine
pbTextUnitObject
pbTextUnitParaFormat
pbTextUnitParagraph
pbTextUnitRow
pbTextUnitScreen
pbTextUnitSection
pbTextUnitSentence
pbTextUnitStory
pbTextUnitTable
pbTextUnitWindow
pbTextUnitWord

Size Required **Long**. The number of units to move. If this number is positive,

the ending character position is moved forward in the document. If this number is negative, the end is moved backward. If the ending position overtakes the starting position, the range collapses and both character positions move together.

Remarks

Use the **MoveEnd** method to move the ending character position for a range.

Example

This example sets a text range, moves the range's starting and ending character positions, and then formats the font for the range.

```
Sub MoveStartEnd()
    Dim rngText As TextRange

    Set rngText = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Paragraphs(Start:=3, Length:=1)

    With rngText
        .MoveStart Unit:=pbTextUnitLine, Size:=-2
        .MoveEnd Unit:=pbTextUnitLine, Size:=1
        With .Font
            .Bold = msoTrue
            .Size = 15
        End With
    End With

End Sub
```

Name Method

Returns a **String** that represents the name of a hidden Web field for a Web command button.

expression.**Name**(***Index***)

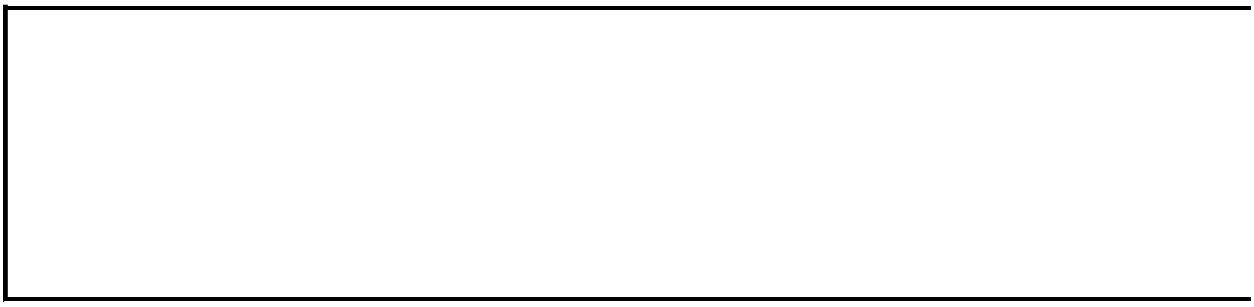
expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Long**. The index number of the hidden field.

Example

This example creates a Web command button with a hidden field, then displays the field's name.

```
Sub GetHiddenWebFieldName()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlCommandButton, _
            Left:=100, Top:=100, Width:=100, _
            Height:=36).WebCommandButton.HiddenFields
        .Add Name:="User", Value:="Power"
        MsgBox "The name of the first hidden field is " & .Name(1)
    End With
End Sub
```




Show All

NewDocument Method

Returns a **Document** object that represents a new publication.

expression.**NewDocument**(***Wizard***, ***Design***)

expression Required. An expression that returns one of the objects in the Applies To list.

Wizard Optional **PbWizard**. The wizard to use to create the new publication.

PbWizard can be one of these PbWizard constants.
pbWizardAdvertisements
pbWizardAirplanes
pbWizardBanners
pbWizardBrochures
pbWizardBusinessCards
pbWizardBusinessForms
pbWizardCalendars
pbWizardCatalogs
pbWizardCertificates
pbWizardEnvelopes
pbWizardFlyers
pbWizardGiftCertificates
pbWizardGreetingCards
pbWizardInvitations
pbWizardJapaneseAdvertisements
pbWizardJapaneseAirplanes
pbWizardJapaneseBanners
pbWizardJapaneseBrochures
pbWizardJapaneseBusinessCards
pbWizardJapaneseBusinessForms
pbWizardJapaneseCalendars

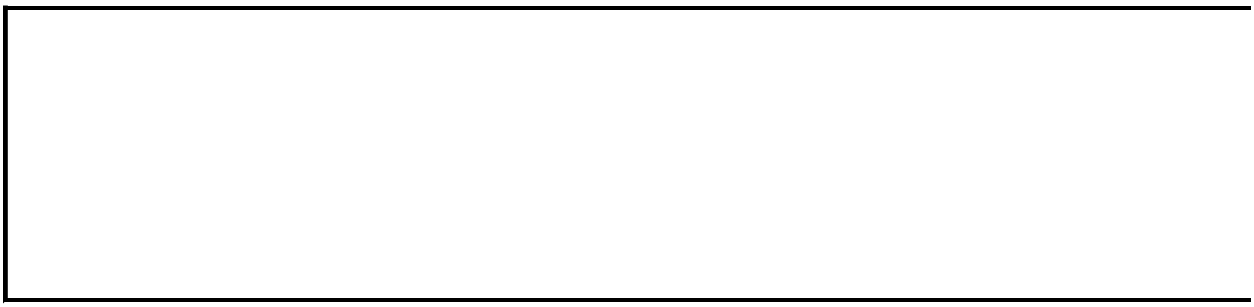
pbWizardJapaneseCatalogs
pbWizardJapaneseCertificates
pbWizardJapaneseEnvelopes
pbWizardJapaneseFlyers
pbWizardJapaneseGiftCertificates
pbWizardJapaneseGreetingCards
pbWizardJapaneseInvitations
pbWizardJapaneseLabels
pbWizardJapaneseLetterheads
pbWizardJapaneseMenus
pbWizardJapaneseNewsletters
pbWizardJapaneseOrigami
pbWizardJapanesePostcards
pbWizardJapanesePrograms
pbWizardJapaneseSigns
pbWizardJapaneseWebSites
pbWizardLabels
pbWizardLetterheads
pbWizardMenus
pbWizardNewsletters
pbWizardNone *default*
pbWizardOrigami
pbWizardPostcards
pbWizardPrograms
pbWizardQuickPublications
pbWizardResumes
pbWizardSigns
pbWizardWebSites
pbWizardWithComplimentsCards
pbWizardWordDocument

Design Optional **Long**. The design to apply to the new publication.

Example

This example creates a new publication and edits the master page to contain a page number in a star in the upper left corner of the page.

```
Sub CreateNewPublication()
    Dim AppPub As Application
    Dim DocPub As Document

    Set AppPub = New Publisher.Application
    Set DocPub = AppPub.NewDocument
    AppPub.ActiveWindow.Visible = True

    With DocPub.MasterPages(1).Shapes.AddShape _
            (Type:=msoShape5pointStar, Left:=36, _
            Top:=36, Width:=50, Height:=50)
        .Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)
        With .TextFrame.TextRange
            .InsertPageNumber
            .ParagraphFormat.Alignment = pbParagraphAlignmentCenter
            With .Font
                .Bold = msoTrue
                .Color.RGB = RGB(Red:=255, Green:=255, Blue:=255)
                .Size = 12
            End With
        End With
    End With
End Sub
```




Show All

OneColorGradient Method

Sets the specified fill to a one-color gradient.

expression.**OneColorGradient**(***Style***, ***Variant***, ***Degree***)

expression Required. An expression that returns one of the objects in the Applies To list.

Style Required **MsoGradientStyle**. The gradient style.

MsoGradientStyle can be one of these MsoGradientStyle constants.
msoGradientDiagonalDown
msoGradientDiagonalUp
msoGradientFromCenter
msoGradientFromCorner
msoGradientFromTitle
msoGradientHorizontal
msoGradientMixed Not used with this method.
msoGradientVertical

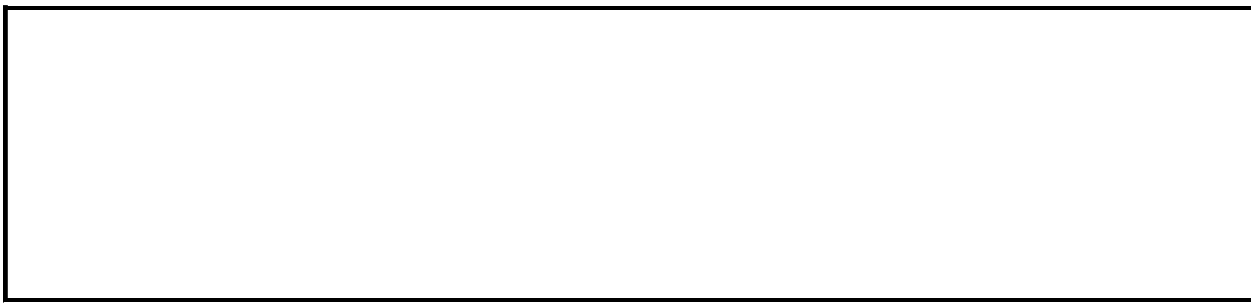
Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

Degree Required **Single**. The gradient degree. Can be a value from 0.0 (dark) to 1.0 (light).

Example

This example adds a rectangle with a one-color gradient fill to the active publication.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeRectangle, _
        Left:=90, Top:=90, Width:=90, Height:=80).Fill
    .ForeColor.RGB = RGB(0, 128, 128)
    .OneColorGradient Style:=msoGradientHorizontal, _
        Variant:=1, Degree:=1
End With
```




Show All

Open Method

Returns a **Document** object that represents the newly opened publication.

expression.**Open**(***FileName***, ***ReadOnly***, ***AddToRecentFiles***, ***SaveChanges***, ***OpenConflictDocument***)

expression Required. An expression that returns one of the objects in the Applies To list.

FileName Required **String**. The name of the publication (paths are accepted).

ReadOnly Optional **Boolean**. **True** to open the publication as read-only. Default is **False**.

AddToRecentFiles Optional **Boolean**. **True** (default) to add the file name to the list of recently used files at the bottom of the **File** menu.

SaveChanges Optional **PbSaveOptions**. Specifies what Publisher should do if there is already an open publication with unsaved changes.

PbSaveOptions can be one of these PbSaveOptions constants.

pbDoNotSaveChanges Close the open publication without saving any changes.

pbPromptToSaveChanges *default* Prompt the user whether to save changes in the open publication.

pbSaveChanges Save the open publication before closing it.

OpenConflictDocument Optional **Boolean**. **True** to open the local conflict publication if there is an offline conflict. Default is **False**.

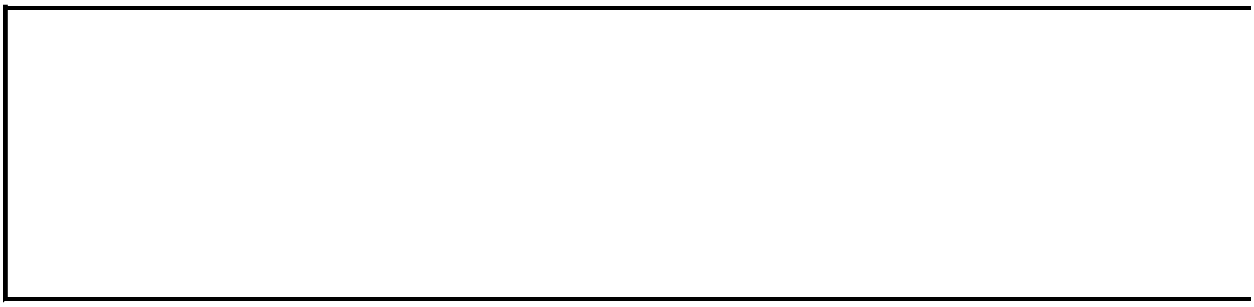
Remarks

Since Publisher has a single document interface, the **Open** method only works when you open a new instance of Publisher. The code sample below shows how to create a new, visible instance of Publisher. When finished with the second instance, you can set the application window's **Visible** property to **False**, but the process continues to run in the background even though it isn't visible. To close the second instance, you must set the object equal to **Nothing**.

Example

This example creates a second instance of Publisher and opens the specified publication as read-only. (Note that *PathToFile* must be replaced with the path to an existing publication for this example to work.)

```
Sub OpenNewPub()
    Dim appPub As New Publisher.Application
    appPub.Open FileName:="PathToFile", _
        ReadOnly:=True, AddToRecentFiles:=False, _
        SaveChanges:=pbPromptToSaveChanges
    appPub.ActiveWindow.Visible = True
End Sub
```



OpenDataSource Method

Attaches a data source to the specified publication, which becomes a main publication if it's not one already.

expression.**OpenDataSource**(***bstrDataSource***, ***bstrConnect***, ***bstrTable***, ***fOpenExclusive***, ***fNeverPrompt***)

expression Required. An expression that returns a **MailMerge** object.

bstrDataSource Optional **String**. The data source path and file name. You can specify a Microsoft Query (.qry) file instead of specifying a data source, a connection string, and a table name string; values in a Microsoft Query file override values for ***bstrConnect*** and ***bstrTable***.

bstrConnect Optional **String**. A connection string.

bstrTable Optional **String**. The name of the table in the data source.

fOpenExclusive Optional **Long**. **True** to deny others access to the database. **False** allows others read/write access to the database. The default value is **False**.

fNeverPrompt Optional **Long**. **True** never prompts when opening the data source. **False** displays the **Data Link Properties** dialog box. The default value is **False**.

Example

This example attaches a table from a database and denies everyone else write access to the database while it is opened. (Note that *PathToFile* must be replaced with a valid file path, and *TableName* with a valid data source table name, for this example to execute properly.)

```
Sub AttachDataSource()

    ActiveDocument.MailMerge.OpenDataSource _
        bstrDataSource:="PathToFile",  _
        bstrTable:="TableName", _
        fNeverPrompt:=True, fOpenExclusive:=True

End Sub
```

OpenRecipientsDialog Method

Displays the **Recipients** dialog box for a mail merge publication.

expression.**OpenRecipientsDialog**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example displays the **Mail Merge Recipients** dialog box.

```
Sub ShowRecipientsDialog()
    ActiveDocument.MailMerge.DataSource.OpenRecipientsDialog
End Sub
```

Paragraphs Method

Returns a **TextRange** object that represents the specified paragraphs.

expression.**Paragraphs**(***Start***, ***Length***)

expression Required. An expression that returns a **TextRange** object.

Start Required **Long**. The first paragraph in the returned range.

Length Optional **Long**. The number of paragraphs to be returned. Default is 1.

Remarks

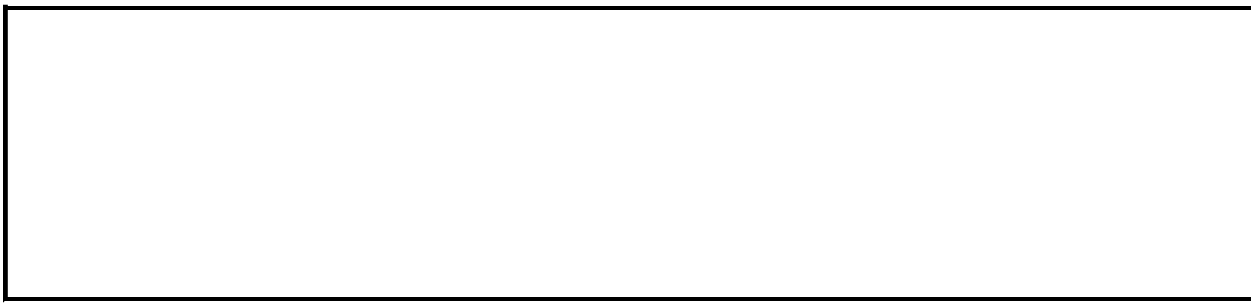
If ***Length*** is omitted, the returned range contains one paragraph.

If ***Length*** is greater than the number of paragraphs from the specified starting paragraph to the end of the text, the returned range contains all those paragraphs.

Example

This example formats as indents the first line of the selected paragraph.

```
Sub FormatCurrentParagraph()
    Selection.TextRange.Paragraphs(Start:=1).ParagraphFormat _
        .FirstLineIndent = InchesToPoints(0.5)
End Sub
```




Show All

Paste Method

 Paste method as it applies to the **Shapes** object.

Pastes the shapes or text on the Clipboard into the specified **Shapes** collection, at the top of the z-order. Each pasted object becomes a member of the specified **Shapes** collection. If the Clipboard contains a text range, the text will be pasted into a newly created **TextFrame** shape. Returns a **ShapeRange** object that represents the pasted objects.

expression.**Paste**

expression Required. An expression that returns a **Shapes** collection.

 Paste method as it applies to the **TextRange** object.

Pastes the text on the Clipboard into the specified text range, and returns a **TextRange** object that represents the pasted text.

expression.**Paste**

expression Required. An expression that returns a **TextRange** object.

Example

 As it applies to the **Shapes** object.

This example copies shape one on page one in the active publication to the Clipboard and then pastes it into page two.

```
With ActiveDocument
    .Pages(1).Shapes(1).Copy
    .Pages(2).Shapes.Paste
End With
```

 As it applies to the **TextRange** object.

This example cuts the text in shape one on page one in the active publication, places it on the Clipboard, and then pastes it after the first word in shape two on the same page. This example assumes that each shape contains text.

```
With ActiveDocument.Pages(1)
    .Shapes(1).TextFrame.TextRange.Cut
    .Shapes(2).TextFrame.TextRange. _
        Words(1).Paste
End With
```


Show All

Patterned Method

Sets the specified fill to a pattern.

expression.**Patterned**(***Pattern***)

expression Required. An expression that returns one of the objects in the Applies To list.

Pattern Required **MsoPatternType**. The pattern to be used for the specified fill.

MsoPatternType can be one of these MsoPatternType constants.
msoPattern5Percent
msoPattern10Percent
msoPattern20Percent
msoPattern25Percent
msoPattern30Percent
msoPattern40Percent
msoPattern50Percent
msoPattern60Percent
msoPattern70Percent
msoPattern75Percent
msoPattern80Percent
msoPattern90Percent
msoPatternDarkDownwardDiagonal
msoPatternDarkHorizontal
msoPatternDarkUpwardDiagonal
msoPatternDarkVertical
msoPatternDashedDownwardDiagonal
msoPatternDashedHorizontal
msoPatternDashedUpwardDiagonal
msoPatternDashedVertical

msoPatternDiagonalBrick
msoPatternDivot
msoPatternDottedDiamond
msoPatternDottedGrid
msoPatternHorizontalBrick
msoPatternLargeCheckerBoard
msoPatternLargeConfetti
msoPatternLargeGrid
msoPatternLightDownwardDiagonal
msoPatternLightHorizontal
msoPatternLightUpwardDiagonal
msoPatternLightVertical
msoPatternMixed Not used with this method.
msoPatternNarrowHorizontal
msoPatternNarrowVertical
msoPatternOutlinedDiamond
msoPatternPlaid
msoPatternShingle
msoPatternSmallCheckerBoard
msoPatternSmallConfetti
msoPatternSmallGrid
msoPatternSolidDiamond
msoPatternSphere
msoPatternTrellis
msoPatternWave
msoPatternWeave
msoPatternWideDownwardDiagonal
msoPatternWideUpwardDiagonal
msoPatternZigZag

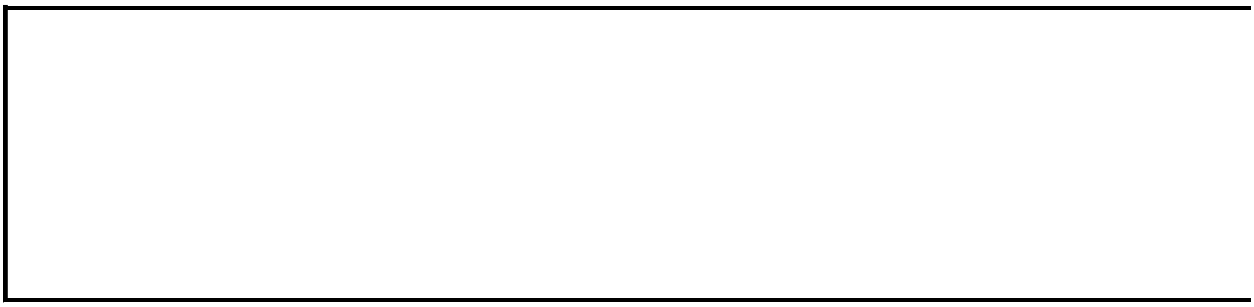
Remarks

Use the **BackColor** and **ForeColor** properties to set the colors used in the pattern.

Example

This example adds an oval with a patterned fill to the active publication.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, _
        Left:=60, Top:=60, Width:=80, Height:=40).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(0, 0, 255)
    .Patterned Pattern:=msoPatternDarkVertical
End With
```



PicasToPoints Method

Converts a measurement from picas to points (1 pica = 12 points). Returns the converted measurement as a **Single**.

expression.**PicasToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The pica value to be converted to points.

Remarks

Use the **PointsToPicas** method to convert measurements in points to picas.

Example

This example converts measurements in picas entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in picas (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " picas = " _
        & Format(Application _
        .Picas ToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```

PickUp Method

Copies formatting from a shape or shape range so that it can be copied to another shape or shape range using the **Apply** method.

expression.**PickUp**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

You must use the **PickUp** method to copy the formatting from a shape or shape range before using the **Apply** method; otherwise, an error occurs.

Example

The following example copies the formatting from the first shape of the active publication to the second shape of the active publication.

```
With ActiveDocument.Pages(1)
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```

PixelsToPoints Method

Converts a measurement from pixels to points (1 pixel = 0.75 points). Returns the converted measurement as a **Single**.

expression.**PixelsToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The pixel value to be converted to points.

Remarks

Use the **PointsToPixels** method to convert measurements in points to pixels.

Example

This example converts measurements in pixels entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in pixels (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " pixels = " _
        & Format(Application _
        .PixelsToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```

PointsToCentimeters Method

Converts a measurement from points to centimeters (1 cm = 28.35 points). Returns the converted measurement as a **Single**.

expression.**PointsToCentimeters**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to centimeters.

Remarks

Use the **CentimetersToPoints** method to convert measurements in centimeters to points.

Example

This example converts measurements in points entered by the user to measurements in centimeters.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToCentimeters(Value:=Val(strInput)), _
        "0.00") & " cm"

    MsgBox strOutput
Loop
```

PointsToEmus Method

Converts a measurement from points to emus (12700 emus = 1 point). Returns the converted measurement as a **Single**.

expression.**PointsToEmus**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to emus.

Remarks

Use the **EmusToPoints** method to convert measurements in emus to points.

Example

This example converts measurements in points entered by the user to measurements in centimeters.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToEmus(Value:=Val(strInput)), _
        "0.00") & " emus"

    MsgBox strOutput
Loop
```

PointsToInches Method

Converts a measurement from points to inches (1 in = 72 points). Returns the converted measurement as a **Single**.

expression.**PointsToInches**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to inches.

Remarks

Use the **InchesToPoints** method to convert measurements in inches to points.

Example

This example converts measurements in points entered by the user to measurements in inches.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToInches(Value:=Val(strInput)), _
        "0.00") & " in"

    MsgBox strOutput
Loop
```

PointsToLines Method

Converts a measurement from points to lines (1 line = 12 points). Returns the converted measurement as a **Single**.

expression.**PointsToLines**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to lines.

Remarks

This method assumes a measurement in 12-point lines— the actual size of any text in the publication has no effect on the conversion factor.

Use the **LinesToPoints** method to convert measurements in lines to points.

Example

This example converts measurements in lines to measurements in points, demonstrating that the font size in the current selection has no bearing on the conversion factor. Some text must be selected in the active publication for this example to work.

```
Dim strOutput As String

' Set text size to 10 points.
Selection.TextRange.Font.Size = 10

' Display result for 12 points.
strOutput = "12 points = " _
    & Format(Application _
    .PointsToLines(Value:=12), _
    "0.00") & " lines"
```

PointsToMillimeters Method

Converts a measurement from points to millimeters (1 mm = 2.835 points). Returns the converted measurement as a **Single**.

expression.**PointsToMillimeters**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to millimeters.

Remarks

Use the **MillimetersToPoints** method to convert measurements in millimeters to points.

Example

This example converts measurements in points entered by the user to measurements in centimeters.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToMillimeters(Value:=Val(strInput)), _
        "0.00") & " mm"

    MsgBox strOutput
Loop
```

PointsToPicas Method

Converts a measurement from points to picas (1 pica = 12 points). Returns the converted measurement as a **Single**.

expression.**PointsToPicas**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to picas.

Remarks

Use the **PicasToPoints** method to convert measurements in picas to points.

Example

This example converts measurements in points entered by the user to measurements in picas.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToPicas(Value:=Val(strInput)), _
        "0.00") & " picas"

    MsgBox strOutput
Loop
```

PointsToPixels Method

Converts a measurement from points to pixels (1 pixel = 0.75 points). Returns the converted measurement as a **Single**.

expression.**PointsToPixels**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to pixels.

Remarks

Use the **PixelsToPoints** method to convert measurements in pixels to points.

Example

This example converts measurements in points entered by the user to measurements in pixels.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToPixels(Value:=Val(strInput)), _
        "0.00") & " pixels"

    MsgBox strOutput
Loop
```

PointsToTwips Method

Converts a measurement from points to twips (20 twips = 1 point). Returns the converted measurement as a **Single**.

expression.**PointsToTwips**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The point value to be converted to twips.

Remarks

Use the **TwipsToPoints** method to convert measurements in twips to points.

Example

This example converts measurements in points entered by the user to measurements in centimeters.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in points (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " points = " _
        & Format(Application _
        .PointsToTwips(Value:=Val(strInput)), _
        "0.00") & " twips"

    MsgBox strOutput
Loop
```


Show All

PresetDrop Method

Specifies whether the callout line attaches to the top, bottom, or center of the callout text box or whether it attaches at a point that's a specified distance from the top or bottom of the text box.

expression.**PresetDrop**(***DropType***)

expression Required. An expression that returns one of the objects in the Applies To list.

DropType Required **MsoCalloutDropType**. The starting position of the callout line relative to the text bounding box.

MsoCalloutDropType can be one of these MsoCalloutDropType constants.

msoCalloutDropBottom

msoCalloutDropCenter

msoCalloutDropCustom

msoCalloutDropMixed Not used with this method.

msoCalloutDropTop

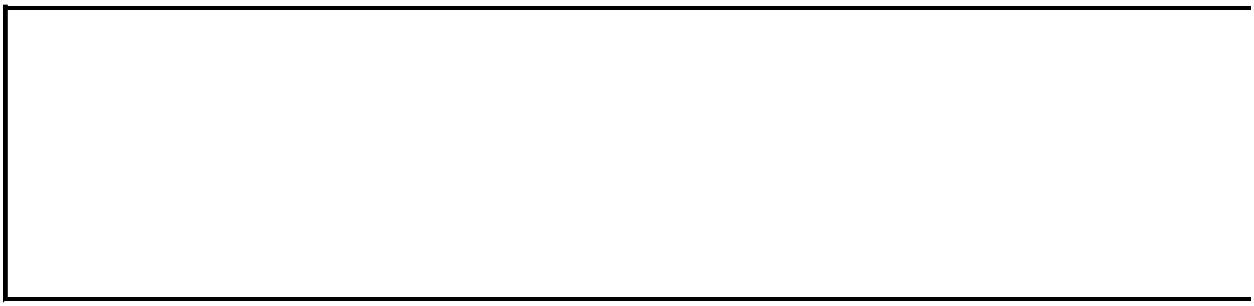
Example

This example specifies that the callout line attach to the top of the text bounding box for the first shape in the active publication. For the example to work, the shape must be a callout.

```
ActiveDocument.Pages(1).Shapes(1).Callout _
    .PresetDrop DropType:=msoCalloutDropTop
```

This example toggles between two preset drops for the first shape one in the active publication. For the example to work, the shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    Select Case .DropType
        Case msoCalloutDropTop
            .PresetDrop DropType:=msoCalloutDropBottom
        Case msoCalloutDropBottom
            .PresetDrop DropType:=msoCalloutDropTop
    End Select
End With
```




Show All

PresetGradient Method

Sets the specified fill to a preset gradient.

expression.**PresetGradient**(***Style***, ***Variant***, ***PresetGradientType***)

expression Required. An expression that returns one of the objects in the Applies To list.

Style Required **MsoGradientStyle**. The style of the gradient.

MsoGradientStyle can be one of these MsoGradientStyle constants.
msoGradientDiagonalDown
msoGradientDiagonalUp
msoGradientFromCenter
msoGradientFromCorner
msoGradientFromTitle
msoGradientHorizontal
msoGradientMixed Not used with this method.
msoGradientVertical

Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

PresetGradientType Required **MsoPresetGradientType**. The gradient type.

MsoPresetGradientType can be one of these MsoPresetGradientType constants.
msoGradientBrass
msoGradientCalmWater
msoGradientChrome
msoGradientChromeII
msoGradientDaybreak

msoGradientDesert
msoGradientEarlySunset
msoGradientFire
msoGradientFog
msoGradientGold
msoGradientGoldII
msoGradientHorizon
msoGradientLateSunset
msoGradientMahogany
msoGradientMoss
msoGradientNightfall
msoGradientOcean
msoGradientParchment
msoGradientPeacock
msoGradientRainbow
msoGradientRainbowII
msoGradientSapphire
msoGradientSilver
msoGradientWheat
msoPresetGradientMixed Not used with this method.

Example

This example adds a rectangle with a preset gradient fill to the active publication.

```
ActiveDocument.Pages(1).Shapes _
    .AddShape(msoShapeRectangle, 90, 90, 140, 80) _
    .Fill.PresetGradient Style:=msoGradientHorizontal, _
    Variant:=1, PresetGradientType:=msoGradientBrass
```


Show All

PresetTextured Method

Sets the specified fill to a preset texture.

expression.**PresetTextured**(***PresetTexture***)

expression Required. An expression that returns one of the objects in the Applies To list.

PresetTexture Required **MsoPresetTexture**. The preset texture.

MsoPresetTexture can be one of these MsoPresetTexture constants.
msoPresetTextureMixed Not used with this method.
msoTextureBlueTissuePaper
msoTextureBouquet
msoTextureBrownMarble
msoTextureCanvas
msoTextureCork
msoTextureDenim
msoTextureFishFossil
msoTextureGranite
msoTextureGreenMarble
msoTextureMediumWood
msoTextureNewsprint
msoTextureOak
msoTexturePaperBag
msoTexturePapyrus
msoTextureParchment
msoTexturePinkTissuePaper
msoTexturePurpleMesh
msoTextureRecycledPaper
msoTextureSand
msoTextureStationery

msoTextureWalnut
msoTextureWaterDroplets
msoTextureWhiteMarble
msoTextureWovenMat

Example

This example adds a rectangle with a green-marble textured fill to the active publication.

```
ActiveDocument.Pages(1).Shapes _
    .AddShape(Type:=msoShapeCan, _
    Left:=90, Top:=90, Width:=40, Height:=80) _
    .Fill.PresetTextured _
    PresetTexture:=msoTextureGreenMarble
```

PrintOut Method

Prints all or part of the specified publication.

expression.**PrintOut**(***From***, ***To***, ***PrintToFile***, ***Copies***, ***Collate***)

expression Required. An expression that returns one of the objects in the Applies To list.

From Optional **Long**. The starting page number.

To Optional **Long**. The ending page number.

PrintToFile Optional **String**. The path and file name of a document to be printed to a file.

Copies Optional **Long**. The number of copies to be printed.

Collate Optional **Boolean**. When printing multiple copies of a document, **True** to print all pages of the document before printing the next copy.

Example

This example prints the active publication.

```
Sub PrintActivePublication()
    ThisDocument.PrintOut
End Sub
```

Quit Method

Quits Microsoft Publisher. This is equivalent to clicking **Exit** on the **File** menu.

expression.**Quit**

expression Required. An expression that returns one of the objects in the Applies To list.

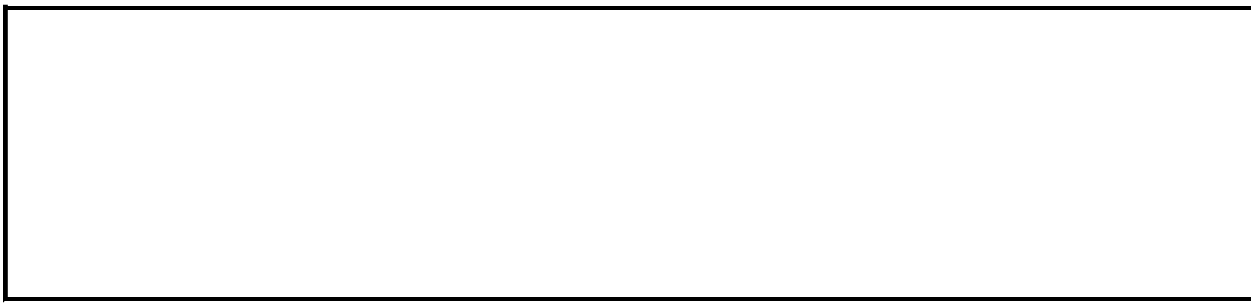
Remarks

To avoid losing unsaved changes, use either the **Save** or **SaveAs** method to save any open publication before calling the **Quit** method.

Example

This example saves the open publication if there is one and then quits Publisher.

```
If Not (ActiveDocument Is Nothing)
    ActiveDocument.Save
End If
Application.Quit
```



Range Method

Returns a **ShapeRange** object that represents a subset of the shapes in a **Shapes** collection.

expression.**Range**(***Index***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Optional **Variant**. The individual shapes that are to be included in the range. Can be an integer that specifies the index number of the shape, a string that specifies the name of the shape, or an array that contains either integers or strings. If ***Index*** is omitted, the **Range** method returns all the objects in the specified collection.

Remarks

To specify an array of integers or strings for ***Index***, you can use the **Array** function. For example, the following instruction returns two shapes specified by name.

```
Dim arrShapes As Variant
Dim shpRange As ShapeRange

Set arrShapes = Array("Oval 4", "Rectangle 5")
Set shpRange = ActiveDocument.Pages(1) _
    .Shapes.Range(arrShapes)
```

Example

This example sets the fill pattern for shapes one and three on the active publication.

```
ActiveDocument.Pages(1).Shapes.Range(Array(1, 3)).Fill _
    .Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for the shapes named "Oval 4" and "Rectangle 5" on the first page.

```
Dim arrShapes As Variant
Dim shpRange As ShapeRange

arrShapes = Array("Oval 4", "Rectangle 5")

Set shpRange = ActiveDocument.Pages(1).Shapes.Range(arrShapes)

shpRange.Fill.Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for all shapes on the first page.

```
ActiveDocument.Pages(1).Shapes _
    .Range.Fill.Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for shape one on the first page.

```
Dim shpRange As ShapeRange

Set shpRange = ActiveDocument.Pages(1).Shapes.Range(1)

shpRange.Fill.Patterned msoPatternHorizontalBrick
```

This example creates an array that contains all the AutoShapes on the first page, uses that array to define a shape range, and then distributes all the shapes in that range horizontally.

```
Dim numShapes As Long
Dim numAutoShapes As Long
Dim autoShpArray As Variant
```

```
Dim intLoop As Integer
Dim shpRange As ShapeRange

With ActiveDocument.Pages(1).Shapes

    numShapes = .Count
    If numShapes > 1 Then

        numAutoShapes = 0
        ReDim autoShpArray(1 To numShapes)

        For intLoop = 1 To numShapes
            If .Item(intLoop).Type = msoAutoShape Then
                numAutoShapes = numAutoShapes + 1
                autoShpArray(numAutoShapes) = .Item(intLoop).Name
            End If
        Next

        If numAutoShapes > 1 Then
            ReDim Preserve autoShpArray(1 To numAutoShapes)
            Set shpRange = .Range(autoShpArray)
            shpRange.Distribute _
                DistributeCmd:=msoDistributeHorizontally, _
                RelativeTo:=False
        End If

    End If

End With
```

Redo Method

Redoes the last action or a specified number of actions. Corresponds to the list of items that appears when you click the arrow beside the **Redo** button on the **Standard** toolbar. Calling this method reverses the **Undo** method.

expression.**Redo**([***Count* = *1***])

expression Required. An expression that returns a **Document** object.

Count Optional **Long**. Specifies the number of actions to be redone. Default is 1, meaning that if omitted, only the last action will be redone.

Remarks

If called when there are no actions on the redo stack, or when ***Count*** is greater than the number of actions that currently reside on the stack, the **Redo** method will redo as many actions as possible and ignore the rest.

The maximum number of actions that can be redone in one call to **Redo** is 20.

Example

The following example uses the **Redo** method to redo a subset of the actions that were undone using the **Undo** method.

Part 1 creates a rectangle that contains a text frame on the fourth page of the active publication. Various font properties are set, and text is added to the text frame. In this case, the text "This font is Courier" is set to 12 point bold Courier font.

Part 2 tests whether the text in the text frame is Verdana font. If not, then the **Undo** method is used to undo the last four actions on the undo stack. The **Redo** method is then used to redo the the first two of the last four actions that were just undone. In this case, the third action (setting the font size) and the fourth action (setting the font to bold) are redone. The font name is then changed to Verdana, and the text is modified.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(4)

' Part 1
With theDoc
    With thePage
        ' Setting the shape creates the first action
        Set theShape = .Shapes.AddShape(msoShapeRectangle, _
            75, 75, 190, 30)
        ' Setting the text range creates the second action
        With theShape.TextFrame.TextRange
             ' Setting the font size creates the third action
             .Font.Size = 12
             ' Setting the font to bold creates the fourth action
             .Font.Bold = msoTrue
             ' Setting the font name creates the fifth action
             .Font.Name = "Courier"
             ' Setting the text creates the sixth action
             .Text = "This font is Courier."
        End With
     End With
```

```
    ' Part 2
    If Not thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Verdan
        .Undo (4)
        With thePage
            With theShape.TextFrame.TextRange
                ' Redo redoes the first two of the four actions that
                theDoc.Redo (2)
                .Font.Name = "Verdana"
                .Text = "This font is Verdana."
            End With
        End With
    End If
End With
```

Regroup Method

Regroups the group that the specified shape range belonged to previously. Returns the regrouped shapes as a single **Shape** object.

expression.**Regroup**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **Regroup** method only restores the group for the first previously grouped shape it finds in the specified **ShapeRange** collection. Therefore, if the specified shape range contains shapes that previously belonged to different groups, only one of the groups will be restored.

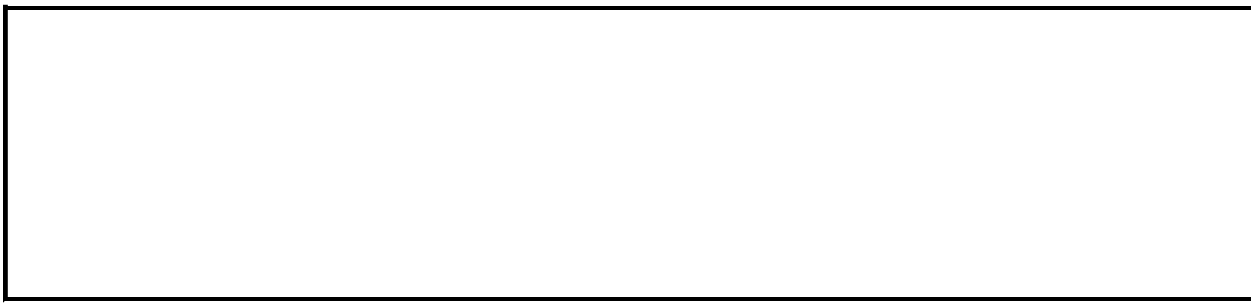
An error occurs if none of the shapes in the specified range were previously members of a group.

Because a group of shapes is treated as a single shape, grouping and ungrouping shapes changes the number of items in the **Shapes** collection and changes the index numbers of items that come after the affected items in the collection.

Example

This example regroups the selected shapes in the active publication. If the shapes haven't been previously grouped and ungrouped, this example will fail.

```
ActiveDocument.Selection.ShapeRange.Regroup
```




Show All

RemoveCatalogMergeArea Method

Deletes the catalog merge area from the specified publication page. All shapes contained in the catalog merge area remain in place on the page, but are no longer connected to the catalog merge data source.

expression.**RemoveCatalogMergeArea**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Removing a catalog merge area from a publication page does not disconnect the data source from the publication. Use the **IsDataSourceConnected** property of the **Document** object to determine if a data source is connected to a publication.

Use the **AddCatalogMergeArea** method of the **Shapes** collection to add a catalog merge area to a publication. A publication page can contain only one catalog merge area.

Example

The following example tests whether any page in the specified publication contains a catalog merge area. If any page does, all the shapes are removed from the catalog merge area and deleted, and the catalog merge area is then removed from the publication.

```
Sub DeleteCatalogMergeAreaAndAllShapesWithin()
    Dim pgPage As Page
    Dim mmLoop As Shape
    Dim intCount As Integer
    Dim strName As String

        For Each pgPage In ThisDocument.Pages
            For Each mmLoop In pgPage.Shapes

                If mmLoop.Type = pbCatalogMergeArea Then
                    With mmLoop.CatalogMergeItems
                        For intCount = .Count To 1 Step -1
                            strName = mmLoop.CatalogMergeItems.Item(
                            .Item(intCount).RemoveFromCatalogMergeAr
                            pgPage.Shapes(strName).Delete
                        Next
                    End With
                mmLoop.RemoveCatalogMergeArea
                End If

            Next mmLoop
        Next pgPage

 End Sub
```


Show All

RemoveFromCatalogMergeArea Method

Removes a shape from the specified page's catalog merge area. Removed shapes are not deleted, but instead remain in place on the page containing the catalog merge area.

expression.**RemoveFromCatalogMergeArea**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **AddToCatalogMergeArea** method of the **Shape** or **ShapeRange** objects to add shapes to a catalog merge area.

Use the **RemoveCatalogMergeArea** method of the **Shape** object to remove the catalog merge area from a publication page, but leave the shapes it contains.

Example

The following example tests whether any page of the specified publication contains a catalog merge area. If any page does, all the shapes are removed from the catalog merge area and deleted, and the catalog merge area is then removed from the publication.

```
Sub DeleteCatalogMergeAreaAndAllShapesWithin()
    Dim pgPage As Page
    Dim mmLoop As Shape
    Dim intCount As Integer
    Dim strName As String

        For Each pgPage In ThisDocument.Pages
            For Each mmLoop In pgPage.Shapes

                If mmLoop.Type = pbCatalogMergeArea Then
                    With mmLoop.CatalogMergeItems
                        For intCount = .Count To 1 Step -1
                            strName = mmLoop.CatalogMergeItems.Item(
                            .Item(intCount).RemoveFromCatalogMergeAr
                            pgPage.Shapes(strName).Delete
                        Next
                    End With
                mmLoop.RemoveCatalogMergeArea
                End If

            Next mmLoop
        Next pgPage

 End Sub
```


Show All

Replace Method

Replaces the specified picture. Returns **Nothing**.

expression.**Replace**(***Pathname***, [***InsertAs***])

expression Required. An expression that returns a **PictureFormat** object.

FileName Required **String**. The name of the file with which you want to replace the specified picture.

InsertAs Optional **PbPictureInsertAs**. The manner in which you want the picture file inserted into the document: linked or embedded.

PbPictureInsertAs can be one of these PbPictureInsertAs constants.
pbPictureInsertAsEmbedded
pbPictureInsertAsLinked
pbPictureInsertAsOriginalState *default*

Remarks

Use the **Replace** method to update linked picture files that have been modified since they were inserted into the document. Use the **LinkedFileStatus** property of the **PictureFormat** object to determine if a linked picture has been modified.

Example

The following example replaces every occurrence of a specific picture in the active publication with another picture.

```
Sub ReplaceLogo()

Dim pgLoop As Page
Dim shpLoop As Shape
Dim strExistingArtName As String
Dim strReplaceArtName As String


strExistingArtName = "C:\pathname\folder\logo 1.bmp"
strReplaceArtName = "C:\pathname\folder\logo 2.bmp"

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat
                If .Filename = strExistingArtName Then
                    .Replace (strReplaceArtName)
                End If
            End With

        End If

    Next shpLoop
Next pgLoop

End Sub
```

This example tests each linked picture to determine if the linked file has been modified since it was inserted into the publication. If it has, the picture is updated by replacing the file with itself.

```
Sub UpdateModifiedLinkedPictures()

Dim pgLoop As Page
Dim shpLoop As Shape
Dim strPictureName As String


For Each pgLoop In ActiveDocument.Pages
```

```
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat
                If .LinkedFileStatus = pbLinkedFileModified Then
                    strPictureName = .Filename
                    .Replace (strPictureName)
                End If
            End With

        End If
    Next shpLoop
Next pgLoop

End Sub
```

RerouteConnections Method

Reroutes connectors so that they take the shortest possible path between the shapes they connect. To do this, the **RerouteConnections** method may detach the ends of a connector and reattach them to different connecting sites on the connected shapes.

expression.**RerouteConnections**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This method reroutes all connectors attached to the specified shape; if the specified shape is a connector, it's rerouted.

Example

This example adds two rectangles to the first page in the active publication and connects them with a curved connector. Note that the **RerouteConnections** method overrides the values you supply for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new curved connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections
    End With

End With
```

Reset Method

Removes manual paragraph or text formatting from the specified object and leaves only the formatting specified by the current text style.

expression.**Reset**

expression Required. An expression that returns one of the objects in the Applies To list.

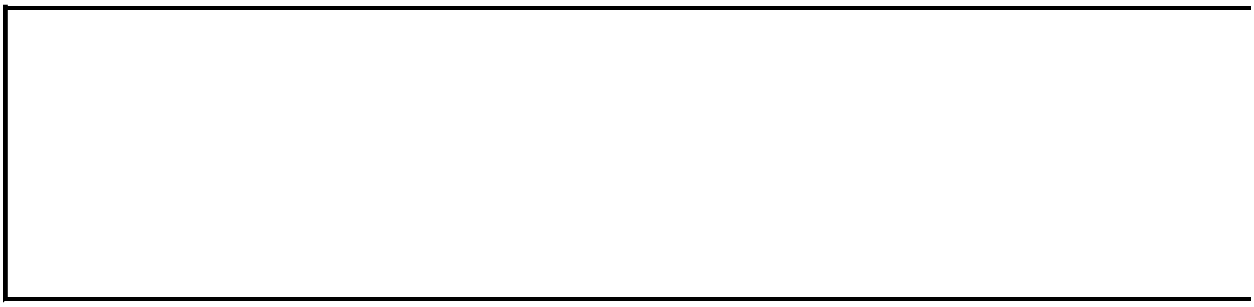
Example

The following example resets the character formatting of the text in shape one on page one of the active publication to the default character formatting for the current text style.

```
ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Font.Reset
```

The following example resets the paragraph formatting of the text in shape one on page one of the active publication to the default paragraph formatting for the current text style.

```
ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.ParagraphFormat.Reset
```



ResetRotation Method

Resets the extrusion rotation around the x-axis (horizontal) and the y-axis (vertical) to 0 (zero) so that the front of the extrusion faces forward.

expression.**ResetRotation**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This method doesn't reset the rotation around the z-axis (extends outward from the plane of the publication).

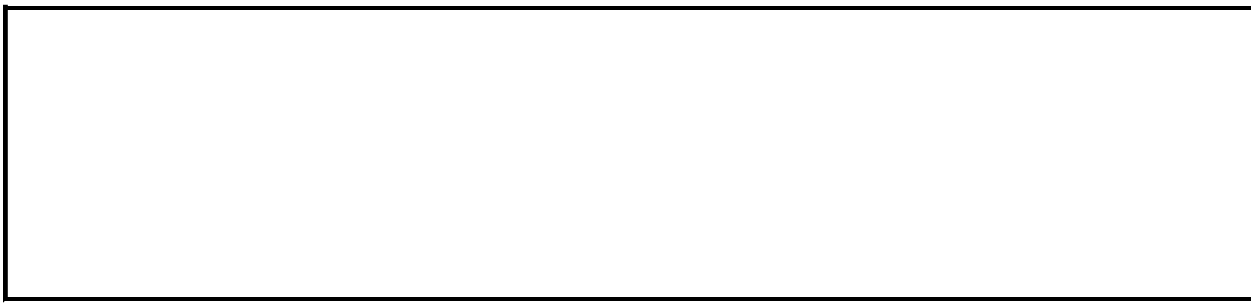
To set the extrusion rotation around the x-axis and the y-axis to anything other than 0, use the **RotationX** and **RotationY** properties of the **ThreeDFormat** object.

To set the extrusion rotation around the z-axis, use the **Rotation** property of the **Shape** object that represents the extruded shape.

Example

This example resets the rotation around the x-axis and the y-axis to zero for the extrusion of the first shape in the active publication.

```
ActiveDocument.Pages(1).Shapes(1).ThreeD _
    .ResetRotation
```



ResetTips Method

Resets tippages so that a user can view them when using features that have been used before.

expression.**ResetTips**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **ResetTips** method is equivalent to clicking **Reset Tips** on the **User Assistance** tab of the **Options** dialog box (**Tools** menu).

Example

This example resets tip balloons.

```
Sub ResetTippages()
    Options.ResetTips
End Sub
```

ResetWizardSynchronizing Method

Resets the data that Microsoft Publisher uses to automatically change similar objects to have the same formatting or content.

expression.**ResetWizardSynchronizing**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Unexpected formatting changes may be a result of Publisher's object synchronization. Resetting the synchronization data will stop these changes.

Example

The following example resets the synchronization data that Publisher uses to give similar objects the same formatting.

```
Options.ResetWizardSynchronizing
```

Resize Method

Sizes the Microsoft Publisher application window.

expression.**Resize**(***Width***, ***Height***)

expression Required. An expression that returns one of the objects in the Applies To list.

Width Required **Long**. The width of the window, in points.

Height Required **Long**. The height of the window, in points.

Remarks

If the window is maximized or minimized, an error occurs.

Use the **Width** and **Height** properties to set the window width and height independently.

Example

This example resizes the Publisher application window to 7 inches wide by 6 inches high.

```
With Application.ActiveWindow
    .WindowState = wdWindowStateNormal
    .Resize Width:=InchesToPoints(7), Height:=InchesToPoints(6)
End With
```

RevertToDefaultWeight Method

Sets the BorderArt on the specified shape back to its default thickness.

expression.**RevertToDefaultWeight**()

expression Required. An expression that returns a **BorderArtFormat** object.

Remarks

The **RevertToDefaultWeight** method has the same effect as the **Always apply at default size** control on the **BorderArt** dialog box.

Use the **Weight** property of the **BorderArtFormat** object to set the specified BorderArt to a thickness other than the default.

Example

The following example tests for the existence of BorderArt on each shape for each page of the active document. If BorderArt exists, its weight is set to the default thickness and original color.

```
Sub RestoreBorderArtDefaults()

Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
        For Each anyShape in anyPage.Shapes
                With anyShape.BorderArt
                        If .Exists = True Then
                                .RevertToDefaultWeight
                                .RevertToOriginalColor
                        End If
                End With
        Next anyShape
Next anyPage
End Sub
```

RevertToOriginalColor Method

Sets the BorderArt on the specified shape back to its default color.

expression.**RevertToOriginalColor**()

expression Required. An expression that returns a **BorderArtFormat** object.

Remarks

The **RevertToOriginalColor** method has the same effect as the **Default** selection on the **Color** control on the **Format <Shape>** dialog box.

Use the **Color** property of the **BorderArtFormat** object to set the BorderArt to a color other than the original color.

Example

The following example tests for the existence of BorderArt on each shape for each page of the active document. If BorderArt exists, its weight is set to the default thickness and original color.

```
Sub RestoreBorderArtDefaults()

Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
        For Each anyShape in anyPage.Shapes
                With anyShape.BorderArt
                        If .Exists = True Then
                                .RevertToDefaultWeight
                                .RevertToOriginalColor
                        End If
                End With
        Next anyShape
Next anyPage
End Sub
```

Save Method

Saves the specified publication.

expression.**Save**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the publication has not been previously saved, calling the **Save** method is equivalent to calling the **SaveAs** method with the ***FileName*** argument set to the value of the publication's **Name** property. If the publication has been previously saved, the **Save** method will save the current version of the publication in the format in which it was opened and in the location to which it was last saved.

Calling the **Save** method always performs saves in the foreground regardless of whether background saves are enabled.

Example

This example saves the active publication if it has changed since it was last saved.

```
If ActiveDocument.Saved = False Then ActiveDocument.Save
```


Show All

SaveAs Method

Saves the specified publication with a new name or format.

expression.**SaveAs**([***FileName***], [***Format***], [***AddToRecentFiles***])

expression Required. An expression that returns one of the objects in the Applies To list.

FileName Optional **Variant**. The name for the publication. The default is the current folder and file name. If the publication has never been saved, the default name is used, for example, Publication1.pub. If a publication with the specified file name already exists, the publication is overwritten without the user being prompted first.

Format Optional **PbFileFormat**. The format in which the publication is saved.

PbFileFormat can be one of these PbFileFormat constants.
pbFileHTMLFiltered
pbFilePublication *default*
pbFilePublicationHTML
pbFilePublisher2000
pbFilePublisher98
pbFileRTF
pbFileWebArchive

AddToRecentFiles Optional **Boolean**. **True** to add the publication to the list of recently used files on the **File** menu. Default is **True**.

Remarks

If there is insufficient memory or disk space to save the file, an error occurs.

Calling the **SaveAs** method always performs saves in the foreground regardless of whether background saves are enabled.

Example

This example saves the active publication as a Publisher 2000 file.

```
ActiveDocument.SaveAs _
    FileName:="ReportPub2000.pub", Format:=pbFilePublisher2000
```

This example saves the active publication as Test.rtf in Rich Text Format (RTF).

```
ActiveDocument.SaveAs _
    FileName:="Test.rtf", Format:=pbFileRTF
```

This example saves the active Web publication as a set of filtered HTML pages and supporting files. Note that the .htm extension is automatically added to the value of the ***Filename*** parameter if the value of the ***Format*** parameter is **pbFileHTMLFiltered**.

```
With ActiveDocument
    .SaveAs Filename:="CompanyContacts", Format:=pbFileHTMLFiltered
End With
```

SaveAsPicture Method

Saves a page to a picture file.

expression.**SaveAsPicture**(***FileName***)

expression Required. An expression that returns one of the objects in the Applies To list.

FileName Required **String**. The path and file name of the new picture created.

Example

This example saves the first page in the active publication as a JPEG picture file. (Note that *PathToFile* must be replaced with a valid file path for this example to execute properly.)

```
Sub SavePageAsPicture()
    ActiveDocument.Pages(1).SaveAsPicture _
        FileName:="PathToFile"
End Sub
```


Show All

ScaleHeight Method

Scales the height of the shape by a specified factor. For pictures and OLE objects, you can indicate whether you want to scale the shape relative to the original size or relative to the current size.

expression.**ScaleHeight**(***Factor***, ***RelativeToOriginalSize***, ***fScale***)

expression Required. An expression that returns one of the objects in the Applies To list.

Factor Required **Single**. Specifies the ratio between the height of the shape after you resize it and the current or original height. For example, to make a rectangle 50 percent larger, specify 1.5 for this argument.

RelativeToOriginalSize Required **MsoTriState**. Specifies whether to scale relative to the object's original or current size.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this method.
msoFalse Scales the shape relative to its current size.
msoTriStateMixed Not used with this method.
msoTriStateToggle Not used with this method.
msoTrue Scales the shape relative to its original size.

fScale Optional **MsoScaleFrom**. The part of the shape that retains its position when the shape is scaled.

MsoScaleFrom can be one of these MsoScaleFrom constants.
msoScaleFromBottomRight
msoScaleFromMiddle
msoScaleFromTopLeft *default*

Remarks

Shapes other than pictures and OLE objects are always scaled relative to their current height; specifying a ***RelativeToOriginalSize*** value of **msoTrue** for shapes other than pictures or OLE objects causes an error.

Use the **ScaleWidth** method to scale the width of a shape.

Example

This example scales all pictures and OLE objects on the first page of the active publication to 175 percent of their original height and width, and it scales all other shapes to 175 percent of their current height and width.

```
' Looping variable.
Dim shpLoop As Shape

' Loop through all the shapes on the first page.
For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop
        Select Case .Type
            ' If the shape is a picture or OLE object,
            ' scale relative to original size.
            Case pbPicture, pbLinkedPicture, _
                    pbEmbeddedOLEObject, pbLinkedOLEObject, _
                    pbOLEControlObject
                .ScaleHeight Factor:=1.75, _
                    RelativeToOriginalSize:=True
                .ScaleWidth Factor:=1.75, _
                    RelativeToOriginalSize:=True
            ' If the shape is not a picture or OLE object,
            ' scale relative to the current size.
            Case Else
                .ScaleHeight Factor:=1.75, _
                    RelativeToOriginalSize:=False
                .ScaleWidth Factor:=1.75, _
                    RelativeToOriginalSize:=False
        End Select
    End With
Next shpLoop
```


Show All

ScaleWidth Method

Scales the width of the shape by a specified factor. For pictures and OLE objects, you can indicate whether you want to scale the shape relative to the original size or relative to the current size.

expression.**ScaleWidth**(***Factor***, ***RelativeToOriginalSize***, ***fScale***)

expression Required. An expression that returns one of the objects in the Applies To list.

Factor Required **Single**. Specifies the ratio between the width of the shape after you resize it and the current or original width. For example, to make a rectangle 50 percent larger, specify 1.5 for this argument.

RelativeToOriginalSize Required **MsoTriState**. Specifies whether to scale relative to the object's original or current size.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this method.
msoFalse Scales the shape relative to its current size.
msoTriStateMixed Not used with this method.
msoTriStateToggle Not used with this method.
msoTrue Scales the shape relative to its original size.

fScale Optional **MsoScaleFrom**. The part of the shape that retains its position when the shape is scaled.

MsoScaleFrom can be one of these MsoScaleFrom constants.
msoScaleFromBottomRight
msoScaleFromMiddle
msoScaleFromTopLeft *default*

Remarks

Shapes other than pictures and OLE objects are always scaled relative to their current width; specifying a ***RelativeToOriginalSize*** value of **msoTrue** for shapes other than pictures or OLE objects causes an error.

Use the **ScaleHeight** method to scale the height of a shape.

Example

This example scales all pictures and OLE objects on the first page of the active publication to 175 percent of their original height and width, and it scales all other shapes to 175 percent of their current height and width.

```
' Looping variable.
Dim shpLoop As Shape

' Loop through all the shapes on the first page.
For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop
        Select Case .Type
            ' If the shape is a picture or OLE object,
            ' scale relative to original size.
            Case pbPicture, pbLinkedPicture, _
                    pbEmbeddedOLEObject, pbLinkedOLEObject, _
                    pbOLEControlObject
                .ScaleHeight Factor:=1.75, _
                    RelativeToOriginalSize:=True
                .ScaleWidth Factor:=1.75, _
                    RelativeToOriginalSize:=True
            ' If the shape is not a picture or OLE object,
            ' scale relative to the current size.
            Case Else
                .ScaleHeight Factor:=1.75, _
                    RelativeToOriginalSize:=False
                .ScaleWidth Factor:=1.75, _
                    RelativeToOriginalSize:=False
        End Select
    End With
Next shpLoop
```

ScrollShapeIntoView Method

Scrolls the publication window so that the specified shape is displayed in the publication window or pane.

expression.**ScrollShapeIntoView**(***Shape***)

expression Required. An expression that returns a **View** object.

Shape Required **Shape** object. The shape to scroll into view.

Example

This example adds a shape to a new page and scrolls the current view to the new shape.

```
Sub ScrollIntoView()
    Dim shpStar As Shape
    Dim intWidth As Integer
    Dim intHeight As Integer

    With ActiveDocument
        intWidth = .PageSetup.PageWidth
        intWidth = (intWidth / 2) - 75
        intHeight = .PageSetup.PageHeight
        intHeight = (intHeight / 2) - 75

        With .Pages.Add(Count:=1, After:=ActiveDocument.Pages.Count)
            Set shpStar = .Shapes.AddShape(Type:=msoShape5pointStar,
                Left:=intWidth, Top:=intHeight, Width:=150, Height:=
            shpStar.TextFrame.TextRange.Text = "New Star Shape"
        End With
    End With

    ActiveView.ScrollShapeIntoView Shape:=shpStar

End Sub
```


Show All

Select Method

 Select method as it applies to the **Cell**, **CellRange**, and **TextRange** objects.

Selects the specified object.

expression.**Select**

expression Required. An expression that returns one of the above objects.

 Select method as it applies to the **Shape** and **ShapeRange** objects.

Selects the specified object.

expression.**Select**(***Replace***)

expression Required. An expression that returns one of the above objects.

Replace Optional **Variant**. Specifies whether the selection replaces any previous selection. **True** to replace the previous selection with the new selection; **False** to add the new selection to the previous selection. Default is **True**.

Example

 As it applies to the **Cell**, **CellRange** and **TextRange** objects.

This example selects the top left cell from a table that has been added to the first page in the active publication.

```
Dim shpTable As Shape
Dim cllTemp As Cell

With ActiveDocument.Pages(1).Shapes
    Set shpTable = .AddTable(NumRows:=3, NumColumns:=3, _
        Left:=100, Top:=100, Width:=150, Height:=150)
    Set cllTemp = shpTable.Table.Cells.Item(1)
    cllTemp.Select
End With
```

This example selects the first column from a table that has been added to the first page in the active publication.

```
Dim shpTable As Shape
Dim crTemp As CellRange

With ActiveDocument.Pages(1).Shapes
    Set shpTable = .AddTable(NumRows:=3, NumColumns:=3, _
        Left:=100, Top:=100, Width:=150, Height:=150)
    Set crTemp = shpTable.Table.Cells(StartRow:=1, _
        StartColumn:=1, EndRow:=3, EndColumn:=1)
    crTemp.Select
End With
```

This example selects the first five characters in shape one on page one of the active publication.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange.Characters(1, 5).Select
```

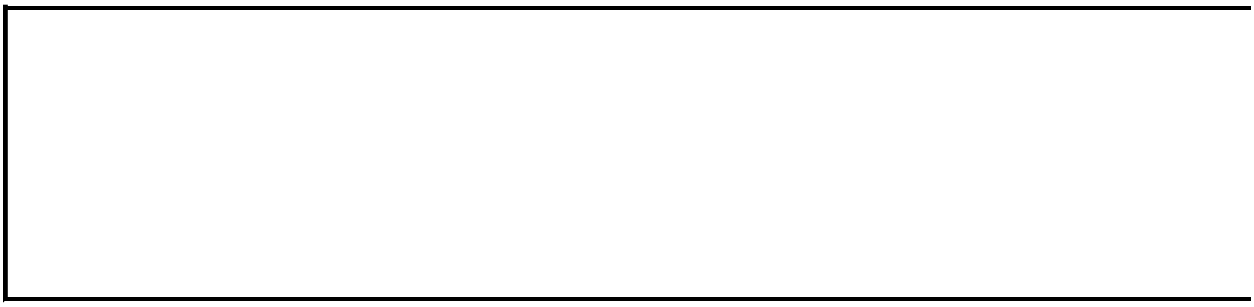
 As it applies to the **Shape** and **ShapeRange** objects.

This example selects shapes one and three on page one in the active publication.

```
ActiveDocument.Pages(1).Shapes.Range(Array(1, 3)).Select
```

This example adds shapes two and four on page one in the active publication to the previous selection.

```
ActiveDocument.Pages(1).Shapes.Range(Array(2, 4)) _
    .Select Replace:=False
```



SelectAll Method

Selects all the shapes in the specified **Shapes** collection.

expression.**SelectAll**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example selects all the shapes on page one of the active publication.

```
ActiveDocument.Pages(1).Shapes.SelectAll
```

Selected Method

Selects or deselects an item in a Web list box control.

expression.**Selected**(***Index***, ***SelectState***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Long**. The number of the Web list box item.

SelectState Required **Boolean**. **True** to select the list item.

Example

This example verifies that an existing Web list box control allows selecting multiple entries and then selects two items in the list.

```
Sub SelectListBoxItem()
    With ActiveDocument.Pages(1).Shapes(1) _
            .WebListBox
        If .MultiSelect = msoTrue Then
            With .ListBoxItems
                .Selected Index:=1, SelectState:=True
                .Selected Index:=3, SelectState:=True
            End With
        End If
    End With
End Sub
```

Set Method

Sets the type of BorderArt applied to the specified shape.

expression.**Set**()

expression Required. An expression that returns one of the objects in the Applies To list.

BorderArtName Required **String**. The name of the BorderArt type applied to the specified picture.

Remarks

You can also set the type of BorderArt applied to a picture using the **Name** property.

Example

The following example tests for the existence of BorderArt on each shape for each page of the active document. Any BorderArt found is set to the same type.

```
Sub SetBorderArt()
Dim anyPage As Page
Dim anyShape As Shape
Dim strBorderArtName As String

strBorderArtName = Document.BorderArts(1).Name

For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Set(strBorderArtName)
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

SetAllErrorFlags Method

Marks all records in a mail merge data source as containing invalid data in an address field.

expression.**SetAllErrorFlags**(***Invalid***, ***InvalidComment***)

expression Required. An expression that returns a **MailMergeDataSource** object.

Invalid Required **Boolean**. **True** marks all records in the data source of a mail merge as invalid.

InvalidComment Optional **String**. Text describing the invalid setting.

Remarks

You can individually mark records in a data source that contain invalid data in an address field using the **InvalidAddress** and **InvalidComments** properties.

Example

This example marks all records in the data source as containing an invalid address field, sets a comment as to why it is invalid, and excludes all records from the mail merge.

```
Sub FlagAllRecords()
    With ActiveDocument.MailMerge.DataSource
        .SetAllErrorFlags Invalid:=True, InvalidComment:= _
            "All records in the data source have only 5-" _
            & "digit ZIP codes.  Need 5+4 digit ZIP codes."
        .SetAllIncludedFlags Included:=False
    End With
End Sub
```

SetAllIncludedFlags Method

True to include all data source records in a mail merge.

expression.**SetAllIncludedFlags**(***Included***)

expression Required. An expression that returns a **MailMergeDataSource** object.

Included Required **Boolean**. **True** to include all data source records in a mail merge. **False** to exclude all data source records from a mail merge.

Remarks

You can set individual records in a data source to be included in or excluded from a mail merge using the **Included** property.

Example

This example marks all records in the data source as containing an invalid address field, sets a comment as to why it is invalid, and excludes all records from the mail merge.

```
Sub FlagAllRecords()
    With ActiveDocument.MailMerge.DataSource
        .SetAllErrorFlags Invalid:=True, InvalidComment:= _
            "All records in the data source have only 5-" _
            & "digit ZIP codes.  Need 5+4 digit ZIP codes."
        .SetAllIncludedFlags Included:=False
    End With
End Sub
```

SetBackgroundSoundRepeat Method

Specifies whether the background sound attached to a Web page should be played infinitely after the page is loaded in a Web browser, and if it should not, optionally specifies the number of times the background sound should be played.

expression.**SetBackgroundSoundRepeat**(***RepeatForever***, [***RepeatTimes***])

expression Required. An expression that returns a **WebPageOptions** object.

RepeatForever Required **Boolean**. Specifies whether the background sound should be played infinitely. The value of this parameter is used to populate the value of the **BackgroundSoundLoopForever** property.

RepeatTimes Optional **Long**. Specifies how many times the background sound should be played. The value of this parameter is used to populate the value of the **BackgroundSoundLoopCount** property.

Remarks

If the ***RepeatForever*** parameter is set to **True**, the optional ***RepeatTimes*** parameter cannot be specified. Attempting to specify ***RepeatTimes*** if ***RepeatForever*** is **True** results in a run-time error.

If the ***RepeatForever*** parameter is set to **False**, the optional ***RepeatTimes*** parameter must be specified. Omitting ***RepeatTimes*** if ***RepeatForever*** is **False** results in a run-time error.

Example

The following example sets the background sound for page four of the active Web publication to a .wav file on the local computer. If **BackgroundSoundLoopForever** is **False**, the **SetBackgroundSoundRepeat** method is used to specify that the background sound be repeated infinitely (note the omission of the ***RepeatTimes*** parameter). If **BackgroundSoundLoopForever** is **True**, the **SetBackgroundSoundRepeat** method is used to specify that the background sound not be repeated infinitely, but that it should be repeated twice.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(4).WebPageOptions

With theWPO
    .BackgroundSound = "C:\CompanySounds\corporate_jingle.wav"
    If .BackgroundSoundLoopForever = False Then
        .SetBackgroundSoundRepeat RepeatForever:=True
    ElseIf .BackgroundSoundLoopForever = True Then
        .SetBackgroundSoundRepeat RepeatForever:=False, RepeatTimes:
    End If

End With
```


Show All

SetCMYK Method

Sets a cyan-magenta-yellow-black (CMYK) color value.

expression.**SetCMYK**(***Cyan***, ***Magenta***, ***Yellow***, ***Black***)

expression Required. An expression that returns a **ColorCMYK** object.

Cyan Required **Long**. A number that represents the cyan component of the color. Value can be any number between 0 and 255.

Magenta Required **Long**. A number that represents the magenta component of the color. Value can be any number between 0 and 255.

Yellow Required **Long**. A number that represents the yellow component of the color. Value can be any number between 0 and 255.

Black Required **Long**. A number that represents the black component of the color. Value can be any number between 0 and 255.

Example

This example sets the CMYK color for the specified shape.

```
Sub SetCMYKColor()
    Dim shpStar As Shape

    Set shpStar = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShape5pointStar, Left:=72, _
        Top:=72, Width:=150, Height:=150)
    shpStar.Fill.ForeColor.CMYK.SetCMYK Cyan:=0, _
        Magenta:=255, Yellow:=255, Black:=50
End Sub
```


Show All

SetEditingType Method

Sets the editing type of the specified node. If the node is a control point for a curved segment, this method sets the editing type of the node adjacent to it that joins two segments. Depending on the editing type, this method may affect the position of adjacent nodes.

expression.**SetEditingType**(***Index***, ***EditingType***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Long**. The node whose editing type is to be set. Must be a number from 1 to the number of nodes in the specified shape; otherwise, an error occurs.

EditingType Required **MsoEditingType**. The editing property of the node.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto Changes the node to a type appropriate to the segments being connected.

msoEditingCorner Changes the node to a corner node.

msoEditingSmooth Changes the node to a smooth curve node.

msoEditingSymmetric Changes the node to a symmetric curve node.

Example

This example changes all corner nodes to smooth nodes in the third shape of the active publication. The shape must be a freeform drawing.

```
Dim intNode As Integer

With ActiveDocument.Pages(1).Shapes(3).Nodes
    For intNode = 1 to .Count
        If .Item(intNode).EditingType = msoEditingCorner Then
            .SetEditingType _
                Index:=intNode, EditingType:=msoEditingSmooth
        End If
    Next intNode
End With
```


Show All

SetExtrusionDirection Method

Sets the direction that the extrusion's sweep path takes away from the extruded shape.

expression.**SetExtrusionDirection**(***PresetExtrusionDirection***)

expression Required. An expression that returns one of the objects in the Applies To list.

PresetExtrusionDirection Required **MsoPresetExtrusionDirection**. Specifies the extrusion direction.

MsoPresetExtrusionDirection can be one of these MsoPresetExtrusionDirection constants.

msoExtrusionBottom

msoExtrusionBottomLeft

msoExtrusionBottomRight

msoExtrusionLeft

msoExtrusionNone

msoExtrusionRight

msoExtrusionTop

msoExtrusionTopLeft

msoExtrusionTopRight

msoPresetExtrusionDirectionMixed Not used with this method.

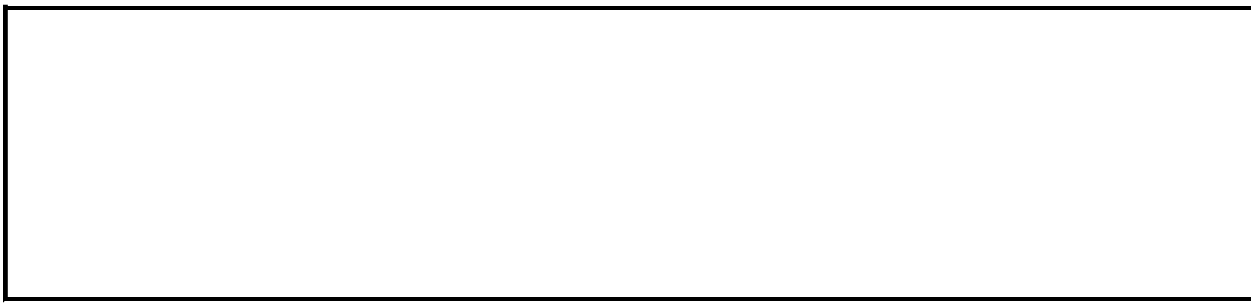
Remarks

This method sets the **PresetExtrusionDirection** property to the direction specified by the ***PresetExtrusionDirection*** argument.

Example

This example specifies that the extrusion for the first shape in the active publication extend toward the top of the shape and that the lighting for the extrusion come from the left.

```
With ActiveDocument.Pages(1).Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection _
        PresetExtrusionDirection:=msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```




Show All

SetLineSpacing Method

Formats the line spacing of specified paragraphs.

expression.**SetLineSpacing**(***Rule***, ***Spacing***)

expression Required. An expression that returns a **ParagraphFormat** object.

Rule Required **PbLineSpacingRule**. The line spacing to use for the specified paragraphs.

PbLineSpacingRule can be one of these PbLineSpacingRule constants.

pbLineSpacing1pt5 Sets the spacing for specified paragraphs to one-and-a-half lines.

pbLineSpacingDouble Double-spaces the specified paragraphs.

pbLineSpacingExactly Sets the line spacing to exactly the value specified in the ***Spacing*** argument, even if a larger font is used within the paragraph.

pbLineSpacingMixed A return value for the **LineSpacing** property that indicates that line spacing is a combination of values for the specified paragraphs.

pbLineSpacingMultiple Sets the line spacing to the value specified in the ***Spacing*** argument.

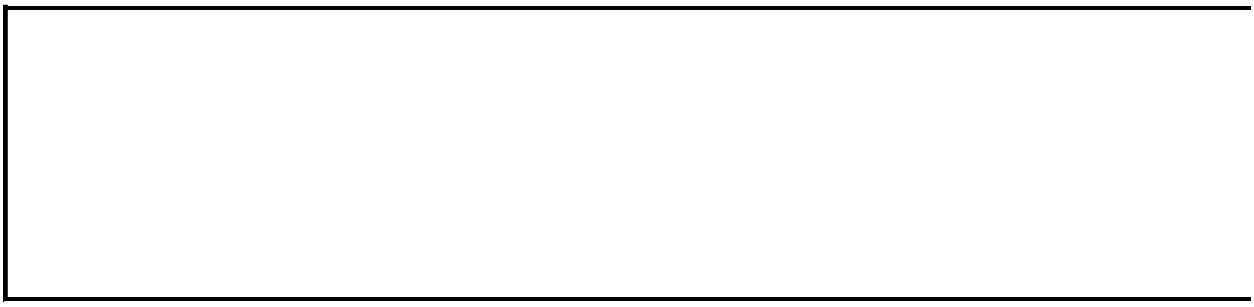
pbLineSpacingSingle Single spaces the specified paragraphs.

Spacing Required **Variant**. The spacing (in points) for the specified paragraphs.

Example

This example sets the line spacing to double.

```
Sub SetLineSpacingForSelection()
    Selection.TextRange.ParagraphFormat.SetLineSpacing _
        Rule:=pbLineSpacingDouble, Spacing:=12
End Sub
```




Show All

SetListType Method

Sets the list type of the specified **ParagraphFormat** object.

expression.**SetListType**(***pbListType***, ***BulletText***)

expression Required. An expression that returns a **ParagraphFormat** object.

PbListType can be one of these **PbListType** constants.
pbListTypeAiueo
pbListTypeArabic
pbListTypeArabic1
pbListTypeArabic2
pbListTypeArabicLeadingZero
pbListTypeBullet
pbListTypeCardinalText
pbListTypeChiManSty
pbListTypeChinaDbNum1
pbListTypeChinaDbNum2
pbListTypeChinaDbNum3
pbListTypeChinaDbNum4
pbListTypeChosung
pbListTypeCirclenum
pbListTypeDAiueo
pbListTypeDArabic
pbListTypeDbChar
pbListTypeDbNum1
pbListTypeDbNum2
pbListTypeDbNum3
pbListTypeDbNum4
pbListTypeDIroha
pbListTypeGanada

pbListTypeGB1
pbListTypeGB2
pbListTypeGB3
pbListTypeGB4
pbListTypeHebrew1
pbListTypeHebrew2
pbListTypeHex
pbListTypeHindi1
pbListTypeHindi2
pbListTypeHindi3
pbListTypeHindi4
pbListTypeIroha
pbListTypeKoreaDbNum1
pbListTypeKoreaDbNum2
pbListTypeKoreaDbNum3
pbListTypeKoreaDbNum4
pbListTypeLowerCaseLetter
pbListTypeLowerCaseRoman
pbListTypeLowerCaseRussian
pbListTypeNone
pbListTypeOrdinal
pbListTypeOrdinalText
pbListTypeSbChar
pbListTypeTaiwanDbNum1
pbListTypeTaiwanDbNum2
pbListTypeTaiwanDbNum3
pbListTypeTaiwanDbNum4
pbListTypeThai1
pbListTypeThai2
pbListTypeThai3
pbListTypeUpperCaseLetter
pbListTypeUpperCaseRoman
pbListTypeUpperCaseRussian

pbListTypeVietnamese1
pbListTypeZodiac1
pbListTypeZodiac2
pbListTypeZodiac3

pbListType Required **pbListType** that represents the list type of the specified **ParagraphFormat** object.

BulletText Optional **String** that represents the text of the list bullet.

Remarks

If the **pbListType** is a bulleted list and the **BulletText** is missing, the first bullet from the **Bullets and Numbering** dialog box is used.

BulletText is limited to one character.

A run-time error occurs if the **BulletText** parameter is provided and the **pbListType** is not set to **pbListTypeBullet**.

Example

This example tests to see if the list type is a numbered list, specifically **pbListTypeArabic**. If the **ListType** property is set to **pbListTypeArabic**, the **ListSeparator** is set to **pbListSeparatorParenthesis**. Otherwise the **SetListType** method is called and passed **pbListTypeArabic** as the **pbListType** parameter and then the **ListNumberSeparator** property can be set.

```
Dim objParaForm As ParagraphFormat

Set objParaForm  = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeArabic Then
        .ListNumberSeparator = pbListSeparatorParenthesis
    Else
        .SetListType pbListTypeArabic
        .ListNumberSeparator = pbListSeparatorParenthesis
    End If
End With
```

This example demonstrates how an organized document structure containing named text frames with lists can be configured. This example assumes that the publication has a naming convention for **TextFrame** objects containing lists that use the word "list" as a prefix. This example uses nested collection iterations to access each of the **TextFrame** objects in each **Shapes** collection of each **Page**. The **ParagraphFormat** object of each **TextFrame** name with the prefix "list" has the **ListType** and **ListBulletFontSize** set.

```
Dim objPage As page
Dim objShp As Shape
Dim objTxtFrm As TextFrame

'iterate through all Pages of Publication
For Each objPage In ActiveDocument.Pages
    'iterate through the Shapes collection of objPage
    For Each objShp In objPage.Shapes
        'find each TextFrame object
        If objShp.Type = pbTextFrame Then
            'if the name of the TextFrame begins with "list"
            If InStr(1, objShp.Name, "list") <> 0 Then
```

```
                Set objTxtFrm = objShp.TextFrame
                With objTxtFrm
                    With .TextRange
                        With .ParagraphFormat
                            .SetListType pbListTypeBullet, "*"
                            .ListBulletFontSize = 24
                        End With
                    End With
                End With
            End If
        End If
    Next
Next
```


Show All

SetPageRelative Method

Sets the target type for the specified hyperlink.

expression.**SetPageRelative**(***RelativePage***)

expression Required. An expression that returns one of the objects in the Applies To list.

RelativePage Required **PbHlinkTargetType**. The target type of the hyperlink.

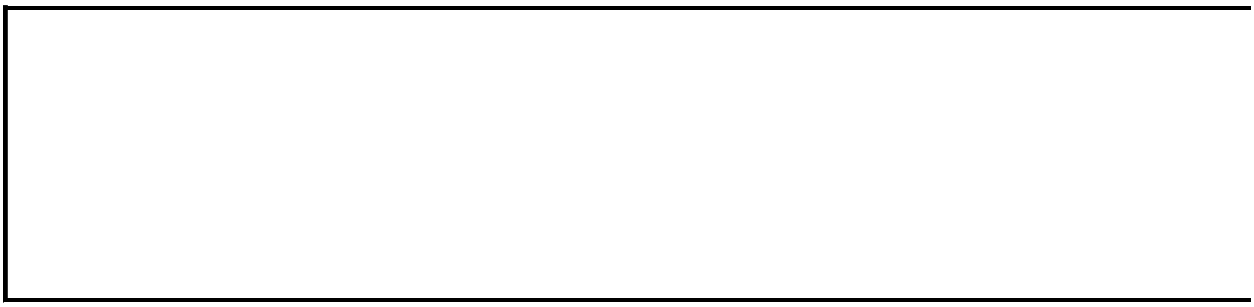
PbHlinkTargetType can be one of these PbHlinkTargetType constants.
pbHlinkTargetTypeEmail
pbHlinkTargetTypeFirstPage
pbHlinkTargetTypeLastPage
pbHlinkTargetTypeNextPage
pbHlinkTargetTypeNone
pbHlinkTargetTypePageID
pbHlinkTargetTypePreviousPage
pbHlinkTargetTypeURL

Example

The following example adds four new hyperlinks to shape one on page one of the active publication and sets their targets accordingly.

```
Sub SetHyperlinkRelativeTarget()
    Dim hypNew As Hyperlink
    Dim txtRng As TextRange

    ActiveDocument.Pages(1).Shapes _
        .AddTextbox Orientation:=pbTextOrientationHorizontal, _
        Left:=10, Top:=10, Width:=200, Height:=200

    Set txtRng = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange

    txtRng.Text = "First Page" & vbCrLf

    Set txtRng = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange
    Set hypNew = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Hyperlinks.Add(Text:=txtRng, _
        Address:="http://www.tailspintoys.com/")

    'Change hyperlink to be a Page-relative link
    hypNew.SetPageRelative RelativePage:=pbHlinkTargetTypeFirstPage

    txtRng.Collapse pbCollapseEnd
    txtRng.Text = "Previous Page" & vbCrLf

    Set hypNew = ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Hyperlinks.Add(Text:=txtRng, _
        Address:="http://www.tailspintoys.com/")

    hypNew.SetPageRelative RelativePage:=pbHlinkTargetTypePreviousPa

    txtRng.Collapse pbCollapseEnd
    txtRng.Text = "Next Page" & vbCrLf
    Set hypNew = ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.Hyperlinks.Add(Text:=txtRng, _
        Address:="http://www.tailspintoys.com/")
    hypNew.SetPageRelative RelativePage:=pbHlinkTargetTypeNextPage

    txtRng.Collapse pbCollapseEnd
    txtRng.Text = "Last Page" & vbCrLf
    Set hypNew = ActiveDocument.Pages(1).Shapes(1) _
```

```
        .TextFrame.TextRange.Hyperlinks.Add(Text:=txtRng, _
        Address:="http://www.tailspintoys.com/")
    hypNew.SetPageRelative RelativePage:=pbHlinkTargetTypeLastPage

End Sub
```



SetPosition Method

Sets the position of the specified node. Depending on the editing type of the node, this method may affect the position of adjacent nodes.

expression.**SetPosition**(***Index***, ***X1***, ***Y1***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Long**. The node whose position is to be set. Must be a number from 1 to the number of nodes in the specified shape; otherwise, an error occurs.

X1 Required **Variant**. The horizontal position of the node relative to the upper-left corner of the page.

Y1 Required **Variant**. The vertical position of the node relative to the upper-left corner of the page.

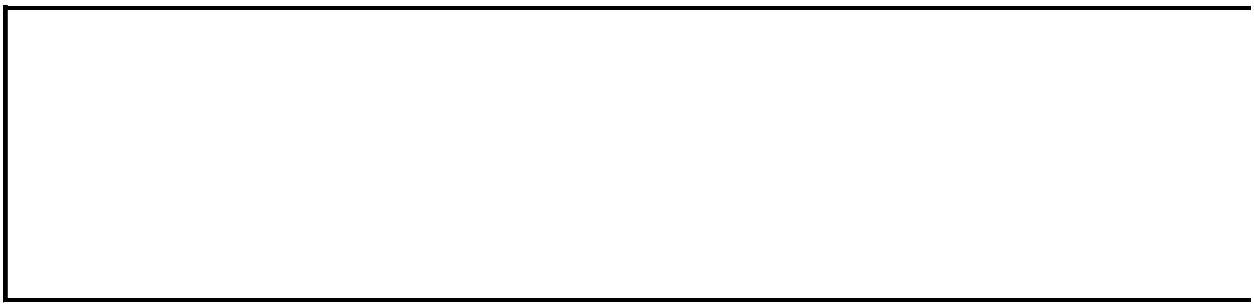
Remarks

For the ***X1*** and ***Y1*** arguments, numeric values are evaluated in points; strings can be in any units supported by Publisher (for example, "2.5 in").

Example

This example moves the second node in the third shape in the active publication 200 points to the right and 300 points down. The shape must be a freeform drawing.

```
Dim arrPoints As Variant
Dim intX As Integer
Dim intY As Integer

With ActiveDocument.Pages(1).Shapes(3).Nodes
    arrPoints = .Item(2).Points
    intX = arrPoints(1, 1)
    intY = arrPoints(1, 2)
    .SetPosition Index:=2, X1:=intX + 200, Y1:=intY + 300
End With
```




Show All

SetScriptName Method

Sets the name of the font script to use in a text range.

expression.**SetScriptName**(***Script***, ***FontName***)

expression Required. An expression that returns one of the objects in the Applies To list.

Script Required **PbFontScriptType**. The script name.

PbFontScriptType can be one of these PbFontScriptType constants.
pbFontScriptArabic
pbFontScriptArmenian
pbFontScriptAsciiLatin
pbFontScriptAsciiSym
pbFontScriptBengali
pbFontScriptBopomofo
pbFontScriptBraille
pbFontScriptCanadianAbor
pbFontScriptCherokee
pbFontScriptCurrency
pbFontScriptCyrillic
pbFontScriptDefault
pbFontScriptDevanagari
pbFontScriptEthiopic
pbFontScriptEUDC
pbFontScriptGeorgian
pbFontScriptGreek
pbFontScriptGujarati
pbFontScriptGurmukhi
pbFontScriptHalfWidthKana
pbFontScriptHan

pbFontScriptHangul
pbFontScriptHanSurrogate
pbFontScriptHebrew
pbFontScriptKana
pbFontScriptKannada
pbFontScriptKhmer
pbFontScriptLao
pbFontScriptLatin
pbFontScriptMalayalam
pbFontScriptMixed
pbFontScriptMongolian
pbFontScriptMyanmar
pbFontScriptNonHanSurrogate
pbFontScriptOgham
pbFontScriptOriya
pbFontScriptRunic
pbFontScriptSinhala
pbFontScriptSyriac
pbFontScriptTamil
pbFontScriptTelugu
pbFontScriptThaana
pbFontScriptThai
pbFontScriptTibetan
pbFontScriptYi

FontName Required **String**. The font name.

Example

This example verifies that the default font script in use for the specified text range is Tahoma and, if not, sets it as the default font script.

```
Sub GetScript()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.Font
        If .GetScriptName(Script:=pbFontScriptDefault) <> "Tahoma" T
            .SetScriptName Script:=pbFontScriptDefault, _
                FontName:="Tahoma"
        End If
    End With
End Sub
```


Show All

SetSegmentType Method

Sets the segment type of the segment that follows the specified node. If the node is a control point for a curved segment, this method sets the segment type for that curve; this may affect the total number of nodes by inserting or deleting adjacent nodes.

expression.**SetSegmentType**(***Index***, ***SegmentType***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Required **Long**. The node whose segment type is to be set. Must be a number from 1 to the number of nodes in the specified shape; otherwise, an error occurs.

SegmentType Required **MsoSegmentType**. Specifies the segment type.

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentCurve

msoSegmentLine

Example

This example changes all straight segments to curved segments in the third shape in the active publication. The shape must be a freeform drawing.

```
Dim intCount As Integer

With ActiveDocument.Pages(1).Shapes(3).Nodes
    intCount = 1
    Do While intCount <= .Count
        If .Item(intCount).SegmentType = msoSegmentLine Then
            .SetSegmentType _
                Index:=intCount, SegmentType:=msoSegmentCurve
        End If
        intCount = intCount + 1
    Loop
End With
```

SetShapesDefaultProperties Method

Applies the formatting for the specified shape or shape range to the default shape. Shapes created after this method has been used will have this formatting applied to them by default.

expression.**SetShapesDefaultProperties**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **SetShapesDefaultProperties** method stores two different sets of default properties, one for a **Shape** object's **AutoShapeType**, and another for a **TextFrame** object. In other words, if this method is called on an AutoShape, the default formatting of that object will apply only to new AutoShapes, and will not apply to new text boxes. If this method is called on a text box, the default formatting of that object will apply only to new text boxes, and will not apply to new AutoShapes.

Example

This example adds a rectangle to the active publication, formats the rectangle's fill, applies the rectangle's formatting to the default shape, and then adds another smaller rectangle to the document. The second rectangle has the same fill as the first one.

```
With ActiveDocument.Pages(1).Shapes

    With .AddShape(Type:=msoShapeRectangle, _
            Left:=5, Top:=5, Width:=80, Height:=60)
        With .Fill
            .ForeColor.RGB = RGB(0, 0, 255)
            .BackColor.RGB = RGB(0, 204, 255)
            .Patterned Pattern:=msoPatternHorizontalBrick
        End With
        .SetShapesDefaultProperties
    End With

    .AddShape Type:=msoShapeRectangle, _
        Left:=90, Top:=90, Width:=40, Height:=30

End With
```

SetSortOrder Method

Sets the sort order for mail merge data.

expression.**SetSortOrder**(***SortField1***, ***SortAscending1***, ***SortField2***, ***SortAscending2***, ***SortField3***, ***SortAscending3***)

expression Required. An expression that returns one of the objects in the Applies To list.

SortField1 Optional **String**. The first field on which to sort the mail merge data. Default is an empty string.

SortAscending1 Optional **Boolean**. **True** (default) to perform an ascending sort on ***SortField1*** ; **False** to perform a descending sort.

SortField2 Optional **String**. The second field on which to sort the mail merge data. Default is an empty string.

SortAscending2 Optional **Boolean**. **True** (default) to perform an ascending sort on ***SortField2*** ; **False** to perform a descending sort.

SortField3 Optional **String**. The third field on which to sort the mail merge data. Default is an empty string.

SortAscending3 Optional **Boolean**. **True** (default) to perform an ascending sort on ***SortField3*** ; **False** to perform a descending sort.

Example

The following example sorts mail merge data first on ZIP code in descending order, then on last name and first name in ascending order.

```
ActiveDocument.MailMerge.DataSource.SetSortOrder _
    SortField1:="ZIPCode", SortAscending1:=False, _
    SortField2:="LastName", SortField3:="FirstName"
```


Show All

SetThreeDFormat Method

Sets the preset extrusion format. Each preset extrusion format contains a set of preset values for the 3-D properties of the extrusion.

expression.**SetThreeDFormat**(***PresetThreeDFormat***)

expression Required. An expression that returns one of the objects in the Applies To list.

PresetThreeDFormat Required **MsoPresetThreeDFormat**. Specifies a preset extrusion format that corresponds to one of the options (numbered from left to right, from top to bottom) displayed when you click the **3-D** button on the **Drawing** toolbar.

MsoPresetThreeDFormat can be one of these MsoPresetThreeDFormat constants.

msoPresetThreeDFormatMixed Not used with this method.

msoThreeD1

msoThreeD2

msoThreeD3

msoThreeD4

msoThreeD5

msoThreeD6

msoThreeD7

msoThreeD8

msoThreeD9

msoThreeD10

msoThreeD11

msoThreeD12

msoThreeD13

msoThreeD14

msoThreeD15

msoThreeD16

msoThreeD17
msoThreeD18
msoThreeD19
msoThreeD20

Remarks

This method sets the **PresetThreeDFormat** property to the format specified by the ***PresetThreeDFormat*** argument.

Example

This example adds an oval to the active publication and sets its extrusion format to one of the preset 3-D formats.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, _
        Left:=30, Top:=30, Width:=50, Height:=25).ThreeD
    .Visible = True
    .SetThreeDFormat PresetThreeDFormat:=msoThreeD12
End With
```

ShowWizard Method

Displays the Mail and Catalog Merge Wizard in a document.

expression.**ShowWizard**(***ShowDocumentStep***, ***ShowTemplateStep***, ***ShowDataStep***, ***ShowWriteStep***, ***ShowPreviewStep***, ***ShowMergeStep***)

expression Required. An expression that returns a **MailMerge** object.

ShowDocumentStep Optional **Boolean**. **True** (default) displays the "Select a merge type" step. **False** removes the step.

ShowTemplateStep Optional **Boolean**. This argument doesn't apply to Publisher.

ShowDataStep Optional **Boolean**. **True** (default) displays the "Select data source" step. **False** removes the step.

ShowWriteStep Optional **Boolean**. **True** (default) displays the "Create your publication" step. **False** removes the step.

ShowPreviewStep Optional **Boolean**. **True** (default) displays the "Preview your publication" step. **False** removes the step.

ShowMergeStep Optional **Boolean**. **True** (default) displays the "Complete the merge" step. **False** removes the step.

Example

This example checks if the Mail Merge Wizard is closed, and if it is, displays it.

```
Sub ShowMergeWizard()
    With ActiveDocument.MailMerge
        If .WizardState = 0 Then
            .ShowWizard
        End If
    End With
End Sub
```

Shrink Method

Decreases the font size to the next available size. If the selection or range contains more than one font size, each size is decreased to the next available setting.

expression.**Shrink**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Applying the **Shrink** method to text that is already the smallest size allowed by Publisher (0.5 point) has no effect.

Example

This example inserts a line of increasingly smaller Z's in a new document.

```
Dim shpText As Shape
Dim trTemp As TextRange
Dim intCount As Integer

Set shpText = ActiveDocument.Pages(1).Shapes _
    .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
    Left:=100, Top:=100, Width:=300, Height:=50)

Set trTemp = shpText.TextFrame.TextRange

With trTemp
    .Font.Size = 45
    .InsertAfter NewText:="ZZZZZZZZZZ"
    For intCount = 2 To 10
        .Characters(Start:=intCount, _
            Length:=11 - intCount).Font.Shrink
    Next intCount
End With
```

Solid Method

Sets the specified fill to a uniform color. Use this method to convert a gradient, textured, patterned, or background fill back to a solid fill.

expression.**Solid**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example converts all fills on the first page of the active publication to uniform red fills.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop.Fill
        .Solid
        .ForeColor.RGB = RGB(255, 0, 0)
    End With
Next shpLoop
```

Split Method

Splits a merged table cell back into its constituent cells. Returns a **CellRange** object representing the constituent cells.

expression.**Split**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the specified cell is not a merged cell resulting from using the **Merge** method, an error occurs.

Example

The following example splits the first cell in the table in shape one on page one of the active publication into its constituent cells. Shape one must contain a table, the first cell of which is a merged cell, in order for this example to work.

```
Dim cllMerged As Cell

Set cllMerged = ActiveDocument.Pages(1) _
    .Shapes(1).Table.Cells.Item(1)

cllMerged.Split
```

ToggleVerticalText Method

Switches the text flow in the specified WordArt from horizontal to vertical, or vice versa.

expression.**ToggleVerticalText**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Using the **ToggleVerticalText** method swaps the values of the **Left** and **Top** properties of the **Shape** object that represents the WordArt and leaves the **Width** and **Height** properties unchanged.

The **Flip** method and **Rotation** property of the **Shape** object and the **RotatedChars** property and **ToggleVerticalText** method of the **TextEffectFormat** object all affect the character orientation and the direction of text flow in a **Shape** object that represents WordArt. You may have to experiment to find out how to combine the effects of these properties and methods to get the result you want.

Example

This example adds WordArt that contains the text "Test" to the active publication, and switches from horizontal text flow (the default for the specified WordArt style, **msoTextEffect1**) to vertical text flow.

```
Dim shpTextEffect As Shape

Set shpTextEffect = ActiveDocument.Pages(1).Shapes.AddTextEffect _
    (PresetTextEffect:=msoTextEffect1, Text:="Test", _
    FontName:="Arial Black", FontSize:=36, _
    FontBold:=False, FontItalic:=False, Left:=100, Top:=100)

shpTextEffect.TextEffect.ToggleVerticalText
```

TwipsToPoints Method

Converts a measurement from twips to points (20 twips = 1 point). Returns the converted measurement as a **Single**.

expression.**TwipsToPoints**(***Value***)

expression Required. An expression that returns one of the objects in the Applies To list.

Value Required **Single**. The twip value to be converted to points.

Remarks

Use the **PointsToTwips** method to convert measurements in points to twips.

Example

This example converts measurements in twips entered by the user to measurements in points.

```
Dim strInput As String
Dim strOutput As String

Do While True
    ' Get input from user.
    strInput = InputBox( _
        Prompt:="Enter measurement in twips (0 to cancel): ", _
        Default:="0")

    ' Exit the loop if user enters zero.
    If Val(strInput) = 0 Then Exit Do

    ' Evaluate and display result.
    strOutput = Trim(strInput) & " twips = " _
        & Format(Application _
        .TwipsToPoints(Value:=Val(strInput)), _
        "0.00") & " points"

    MsgBox strOutput
Loop
```


Show All

TwoColorGradient Method

Sets the specified fill to a two-color gradient. The two fill colors are specified by the **ForeColor** and **BackColor** properties.

expression.**TwoColorGradient**(***Style***, ***Variant***)

expression Required. An expression that returns one of the objects in the Applies To list.

Style Required **MsoGradientStyle**. The gradient style.

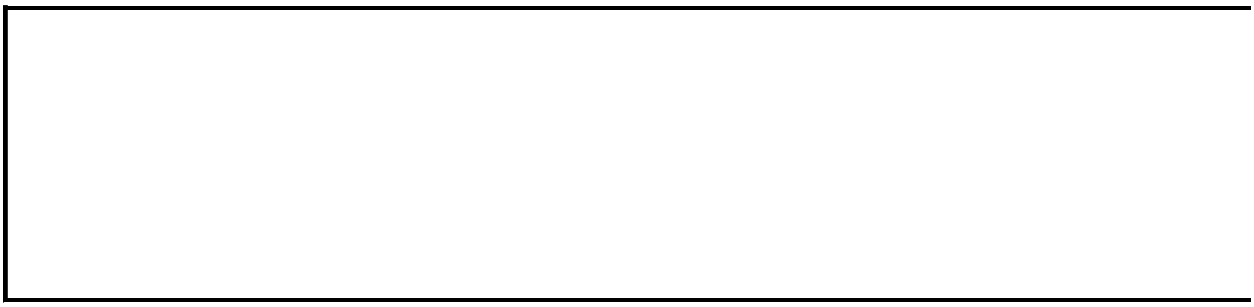
MsoGradientStyle can be one of these MsoGradientStyle constants.
msoGradientDiagonalDown
msoGradientDiagonalUp
msoGradientFromCenter
msoGradientFromCorner
msoGradientFromTitle
msoGradientHorizontal
msoGradientMixed Not used with this method.
msoGradientVertical

Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

Example

This example adds a rectangle with a two-color gradient fill to the active publication and sets the background and foreground color for the fill.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeRectangle, _
        Left:=0, Top:=0, Width:=40, Height:=80).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(0, 170, 170)
    .TwoColorGradient Style:=msoGradientHorizontal, Variant:=1
End With
```



Undo Method

Undoes the last action or a specified number of actions. Corresponds to the list of items that appears when you click the arrow beside the **Undo** button on the **Standard** toolbar.

expression.**Undo**([***Count* = *1***])

expression Required. An expression that returns a **Document** object.

Count Optional **Long**. Specifies the number of actions to be undone. Default is 1, meaning that if omitted, only the last action will be undone.

Remarks

If called when there are no actions on the undo stack, or when ***Count*** is greater than the number of actions that currently reside on the stack, the **Undo** method will undo as many actions as possible and ignore the rest.

The maximum number of actions that can be undone in one call to **Undo** is 20.

Example

The following example uses the **Undo** method to undo actions that do not meet specific criteria.

Part 1 of the example adds a rectangular callout shape to the fourth page of the active publication, and text is added to the callout. This process creates three actions.

Part 2 of the example tests whether the font of the text added to the callout is Verdana. If not, then the **Undo** method is used to undo all available actions (the value of the **UndoActionsAvailable** property is used to specify that all actions be undone). This clears all actions from the stack. A new rectangle shape and text frame are then added and the text frame is populated with Verdana text.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(4)

With theDoc
    ' Part 1
    With thePage
        ' Setting the shape creates the first action
        Set theShape = .Shapes.AddShape(msoShapeRectangularCallout,
                75, 75, 120, 30)
        ' Setting the text range creates the second action
        With theShape.TextFrame.TextRange
             ' Setting the text creates the third action
             .Text = "This text is not Verdana."
        End With
     End With

                                ' Part 2
    If Not thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Verdan
        ' UndoActionsAvailable = 3
        .Undo (.UndoActionsAvailable)
        With thePage
            Set theShape = .Shapes.AddShape(msoShapeRectangle, _
                    75, 75, 120, 30)
            With theShape.TextFrame.TextRange
                .Font.Name = "Verdana"
```

```
                .Text = "This text is Verdana."
            End With
        End With
    End If
End With
```

UndoClear Method

Clears the list of actions that can be undone for the specified publication. Corresponds to the list of items that appears when you click the arrow beside the **Undo** button on the **Standard** toolbar.

expression.**UndoClear**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Include this method at the end of a macro to keep Visual Basic actions from appearing in the **Undo** box (for example, "VBA-Selection.InsertAfter").

Example

This example clears the list of actions that can be undone for the active publication.

```
ActiveDocument.UndoClear
```

Ungroup Method

Ungroups the specified group of shapes or any groups of shapes in the specified shape range. If the specified shape is a picture or OLE object, Microsoft Publisher will break it apart and convert it to an ungrouped set of shapes. (For example, an embedded Microsoft Excel spreadsheet is converted into lines and text boxes.) Returns the ungrouped shapes as a single **ShapeRange** object.

expression.**Ungroup**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Using this method on an inline shape or a shape that isn't a group, picture, or OLE object generates an error. Also, an error occurs if the picture is a bitmap, JPEG, GIF, or PNG (Portable Network Graphics) file.

Because a group of shapes is treated as a single object, grouping and ungrouping shapes changes the number of items in the **Shapes** collection and changes the index numbers of items that come after the affected items in the collection. Also, newly ungrouped shapes are added to the **Shapes** collection on the current page (or pages) or scratch area. As a result, they may shift from one collection to another.

Example

This example ungroups any grouped shapes on the first page of the active publication.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    If shpLoop.Type = pbGroup Then _
        shpLoop.Ungroup
Next shpLoop
```

Unlink Method

Replaces the specified field or **Fields** collection with with their most recent results.

expression.**Unlink**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

When you unlink a field, its current result is converted to text or a graphic and can no longer be updated automatically.

Example

This example unlinks the first field in shape one on the first page of the active publication.

```
ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Fields(1).Unlink
```

This example updates and unlinks all the fields in shape one on the first page of the active publication.

```
With ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.Fields
    .Update
    .Unlink
End With
```

Unselect Method

Cancels the current selection.

expression.**Unselect**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example cancels the current selection in the active publication.

```
ActiveDocument.Selection.Unselect
```

Update Method

Updates the specified linked OLE object.

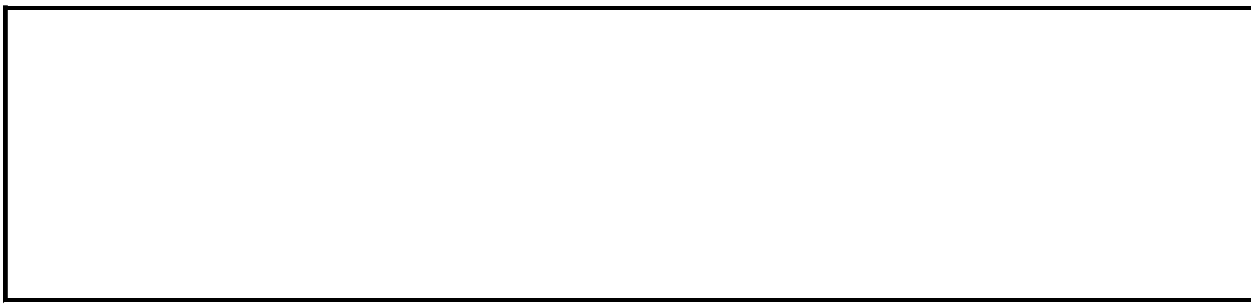
expression.**Update**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example updates all linked OLE objects in the active publication.

```
Dim pageLoop As Page
Dim shpLoop As Shape

For Each pageLoop In ActiveDocument.Pages
    For Each shpLoop In pageLoop.Shapes

        With shpLoop
            If .Type = pbLinkedOLEObject Then
                .LinkFormat.Update
            End If
        End With

    Next shpLoop
Next pageLoop
```



UpdateOLEObjects Method

Updates linked and embedded OLE objects.

expression.**UpdateOLEObjects**

expression Required. An expression that returns a **Document** object.

Example

This example updates all OLE objects in the active publication.

```
Sub SearchAndUpdateOLEObjects()
    ActiveDocument.UpdateOLEObjects
End Sub
```

UserPicture Method

Fills the specified shape with one large image.

expression.**UserPicture**(***PictureFile***)

expression Required. An expression that returns one of the objects in the Applies To list.

PictureFile Required **String**. The name of the picture file.

Remarks

To fill the shape with small tiles of an image, use the **UserTextured** method.

Example

This example adds two rectangles to the active publication. The rectangle on the left is filled with one large image of a picture; the rectangle on the right is filled with many small tiles of the same picture. (Note that *PathToFile* must be replaced with a valid file path for this example to work.)

```
With ActiveDocument.Pages(1).Shapes
    .AddShape(Type:=msoShapeRectangle, _
        Left:=0, Top:=0, Width:=200, Height:=100).Fill _
        .UserPicture PictureFile:="PathToFile"
    .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=0, Width:=200, Height:=100).Fill _
        .UserTextured TextureFile:="PathToFile"
End With
```

UserTextured Method

Fills the specified shape with small tiles of an image.

expression.**UserTextured**(***TextureFile***)

expression Required. An expression that returns one of the objects in the Applies To list.

TextureFile Required **String**. The name of the texture file.

Remarks

To fill the shape with one large image, use the **UserPicture** method.

Example

This example adds two rectangles to the active publication. The rectangle on the left is filled with one large image of a picture; the rectangle on the right is filled with many small tiles of the same picture. (Note that *PathToFile* must be replaced with a valid file path for this example to work.)

```
With ActiveDocument.Pages(1).Shapes
    .AddShape(Type:=msoShapeRectangle, _
        Left:=0, Top:=0, Width:=200, Height:=100).Fill _
        .UserPicture PictureFile:="PathToFile"
    .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=0, Width:=200, Height:=100).Fill _
        .UserTextured TextureFile:="PathToFile"
End With
```

ValidLinkTarget Method

Determines whether the text frame of one shape can be linked to the text frame of another shape. Returns **True** if ***LinkTarget*** is a valid target, **False** if ***LinkTarget*** already contains text or is already linked, or if the shape doesn't support attached text.

expression.ValidLinkTarget(***LinkTarget***)

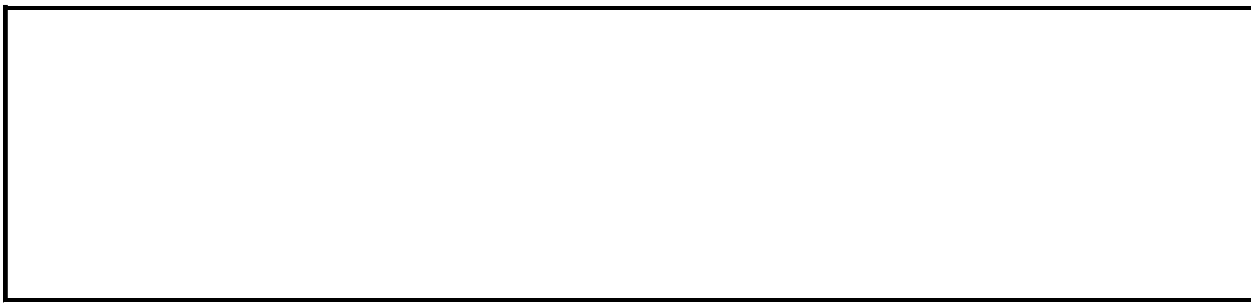
expression Required. An expression that returns one of the objects in the Applies To list.

LinkTarget Required **Shape** object. The shape with the target text frame to which you wish to link the text frame returned by *expression* .

Example

This example checks to see whether the text frames for the first and second shapes on the first page of the active publication can be linked to one another. If so, the example links the two text frames.

```
Dim txtFrame1 As TextFrame
Dim txtFrame2 As TextFrame

With ActiveDocument.Pages(1)
    Set txtFrame1 = .Shapes(1).TextFrame
    Set txtFrame2 = .Shapes(2).TextFrame
End With

If txtFrame1.ValidLinkTarget(LinkTarget:=txtFrame2.Parent) = True Th
    txtFrame1.NextLinkedTextFrame = txtFrame2
End If
```



WebPagePreview Method

Generates a Web page preview of the specified publication in Internet Explorer.

expression.**WebPagePreview**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

A Web preview can be generated for print publications. However, the appearance of the Web preview may differ from the printed publication.

The Web preview opens with the active page displayed. Preview Web pages are generated for each page in the publication. However, if the publication is a print publication or otherwise lacks a navigation bar, there may be no way to navigate to those pages.

Use the **PublicationType** property to determine if a publication is a print publication or a Web publication.

This method corresponds to the **Web Page Preview** command on the **File** menu.

Example

The following example sets the active page of the publication and generates a Web preview of the publication.

```
With ActiveDocument
    .ActiveView.ActivePage = .Pages(2)
    .WebPagePreview
End With
```

Words Method

Returns a **TextRange** object that represents the specified subset of text words.

expression.**Words**(***Start***, ***Length***)

expression Required. An expression that returns one of the objects in the Applies To list.

Start Required **Long**. The first word in the returned range.

Length Optional **Long**. The number of words to be returned. Default is 1.

Remarks

If ***Length*** is omitted, the returned range contains one word.

If ***Start*** is greater than the number of words in the specified text, the returned range starts with the last word in the specified range.

If ***Length*** is greater than the number of words from the specified starting word to the end of the text, the returned range contains all those words.

Example

This example formats as bold the second, third, and fourth words in shape two on page one of the active publication.

```
Application.ActiveDocument.Pages(1).Shapes(2) _
    .TextFrame.TextRange.Words(Start:=2, Length:=3) _
    .Font.Bold = True
```

ZoomIn Method

Increases the magnification of the specified view.

expression.**ZoomIn**

expression Required. An expression that returns a **View** object.

Example

This example increases the magnification of the active view.

```
Sub Zoom()
    ActiveView.ZoomIn
End Sub
```

ZoomOut Method

Decreases the magnification of the specified view.

expression.**ZoomOut**

expression Required. An expression that returns a **View** object.

Example

This example decreases the magnification of the active view.

```
Sub Zoom()
    ActiveView.ZoomOut
End Sub
```


Show All

ZOrder Method

Moves the specified shape in front of or behind other shapes in the collection (that is, changes the shape's position in the z-order).

expression.**ZOrder**(***ZOrderCmd***)

expression Required. An expression that returns one of the objects in the Applies To list.

ZOrderCmd Required **MsoZOrderCmd**. Specifies where to move the specified shape relative to the other shapes.

MsoZOrderCmd can be one of these MsoZOrderCmd constants.

msoBringForward
msoBringInFrontOfText
msoBringToFront
msoSendBackward
msoSendBehindText
msoSendToBack

Remarks

Use the **ZOrderPosition** property to determine a shape's current position in the z-order.

Example

This example adds an oval to the active publication and then places the oval second from the back in the z-order if there is at least one other shape on the page.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, _
        Left:=100, Top:=100, Width:=100, Height:=300)
    While .ZOrderPosition > 2
        .ZOrder ZOrderCmd:=msoSendBackward
    Wend
End With
```


Show All

Accent Property

Returns or sets an **MsoTriState** constant indicating whether a vertical accent bar separates the callout text from the callout line. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse A vertical accent bar does not separate the callout text from the callout line.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Set value only; toggles between **msoTrue** and **msoFalse**.

msoTrue A vertical accent bar separates the callout text from the callout line.

expression.**Accent**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active publication and a callout that points to the oval. The callout text won't have a border, but it will have a vertical accent bar that separates the text from the callout line.

```
With ActiveDocument.Pages(1).Shapes
    ' Add an oval.
    .AddShape Type:=msoShapeOval, _
        Left:=180, Top:=200, Width:=280, Height:=130

    ' Add a callout.
    With .AddCallout(Type:=msoCalloutTwo, _
            Left:=420, Top:=170, Width:=170, Height:=40)

        ' Add text to the callout.
        .TextFrame.TextRange.Text = "This is an oval"

        ' Add an accent bar to the callout.
        With .Callout
            .Accent = msoTrue
            .Border = msoFalse
        End With
    End With
End With
```

ActionURL Property

Returns or sets a **String** that represents the URL of the server-side script to execute in response to a Submit button click. Read/write.

expression.**ActionURL**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The default value for the **ActionURL** property is "http://example.microsoft.com/~user/ispscript.cgi". This property is ignored for Reset command buttons.

Example

This example creates a Web form Submit command button and sets the script path and file name to run when a user clicks the button.

```
Sub CreateActionWebButton()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlCommandButton, Left:=150, _
            Top:=150, Width:=75, Height:=36).WebCommandButton
        .ButtonText = "Submit"
        .ButtonType = pbCommandButtonSubmit
        .ActionURL = "http://www.tailspintoys.com/" & _
            "scripts/ispscript.cgi"
    End With
End Sub
```

ActiveDocument Property

Returns a **Document** object that represents the active publication. If there are no documents open, an error occurs.

expression.**ActiveDocument**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example allows the user to assign a file name to the active publication and save it with the new file name. The file name, along with other text, is then inserted after the currently selected text. (Note that *Filename* must be replacedvin with a valid publication name for this example to work.)

```
Sub NewsLetterSave()

    Dim strFileName As String

    ' Assign the explicit file name to a variable.
    strFileName = "Filename"
    Publisher.ActiveDocument.SaveAs strFileName

    ' Insert the file name and supporting text after selected text.
    Selection.TextRange.Collapse pbCollapseEnd
    Selection.TextRange = _
        " This publication has been saved as " & strFileName

End Sub
```

ActivePage Property

Returns a **Page** object that represents the page currently displayed in the Publisher window.

expression.**ActivePage**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example saves the active page as a JPEG picture. (Note that *PathToFile* must be replaced with a valid file path for this example to work.)

```
Sub SavePageAsPicture()
    ActiveView.ActivePage.SaveAsPicture _
        FileName:="PathToFile"
End Sub
```

This example adds a horizontal and a vertical ruler guide to the active page that intersects at the center point of the page.

```
Sub SetRulerGuidesOnActivePage()
    Dim intHeight As Integer
    Dim intWidth As Integer

    With ActiveView.ActivePage
        intHeight = .Height / 2
        intWidth = .Width / 2
        With .RulerGuides
            .Add Position:=intHeight, Type:=pbRulerGuideTypeHorizont
            .Add Position:=intWidth, Type:=pbRulerGuideTypeVertical
        End With
    End With
End Sub
```

ActivePrinter Property

Returns or sets a **String** corresponding to the name of the active printer. The **ActivePrinter** name is the same string name used to represent the printer in the user interface. Read/write.

expression.**ActivePrinter**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the name of the active printer.

```
MsgBox "The name of the active printer is " & _
    Application.ActiveDocument.ActivePrinter
```

This example makes a network HP LaserJet IIISi printer the active printer.

```
Application.ActiveDocument.ActivePrinter = _

"HP LaserJet IIISi on \\printers\laser"
```

This example makes a local HP LaserJet 4 printer on LPT1 the active printer.

```
Application.ActiveDocument.ActivePrinter = _
    "HP LaserJet 4 local on LPT1:"
```

ActiveRecord Property

Returns or sets a **Long** that represents the active mail merge data record. Read/write.

expression.**ActiveRecord**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The active data record number is the position of the record in the query result produced by the current query options; as such, this number isn't necessarily the position of the record in the data source.

Example

This example validates that the value entered into the PostalCode field is ten characters long (U.S. ZIP code plus 4-digit locator code). If it isn't, it is excluded from the mail merge and marked with a comment.

```
Sub ValidateZip()

    Dim intCount As Integer

    On Error Resume Next

    With ActiveDocument.MailMerge.DataSource

        'Set the active record equal to the first included
        'record in the data source
        .ActiveRecord = 1
        Do
            intCount = intCount + 1

            'Set the condition that the PostalCode field
            'must be greater than or equal to ten digits
            If Len(.DataFields.Item("PostalCode").Value) < 10 Then

                'Exclude the record if the PostalCode field
                'is less than ten digits
                .Included = False

                'Mark the record as containing an invalid address fi
                .InvalidAddress = True

                'Specify the comment attached to the record explaini
                'why the record was excluded from the mail merge
                .InvalidComments = "The ZIP code for this record is
                    & "less than ten digits. It will be removed " _
                    & "from the mail merge process."

            End If

            'Move the record to the next record in the data source
            .ActiveRecord = .ActiveRecord + 1

        'End the loop when the counter variable
        'equals the number of records in the data source
        Loop Until intCount = .RecordCount
    End With
```

```
End Sub
```

ActiveView Property

Returns a **View** object representing the view attributes for the specified document. Read-only.

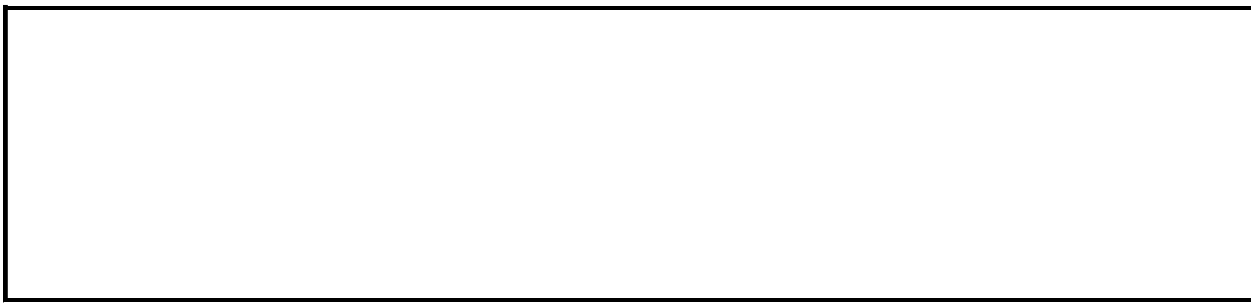
expression.**ActiveView**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets the active publication zoom to fill the screen.

```
Sub SetActiveZoom()
    Dim viewTemp As View

    ActiveDocument.Pages(1).Shapes.AddShape 1, 10, 10, 50, 50
    Set viewTemp = ActiveDocument.ActiveView
    ActiveDocument.Pages(1).Shapes(1).Select
    viewTemp.Zoom = pbZoomFitSelection
End Sub
```



ActiveWindow Property

Returns a **Window** object that represents the window with the focus. Because Microsoft Publisher only has one window, there is only one **Window** object to return.

expression.**ActiveWindow**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the active window's caption.

```
Sub CurrentCaption()

    MsgBox ActiveDocument.ActiveWindow.Caption

End Sub
```

AddHebDoubleQuote Property

True for Publisher to display double quotes for Hebrew alphabet numbering. Default is **False**. Read/write **Boolean**.

expression.**AddHebDoubleQuote**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is only accessible if Hebrew has been enabled for Microsoft Office on your computer.

This property only applies to Hebrew alphabetic numbering.

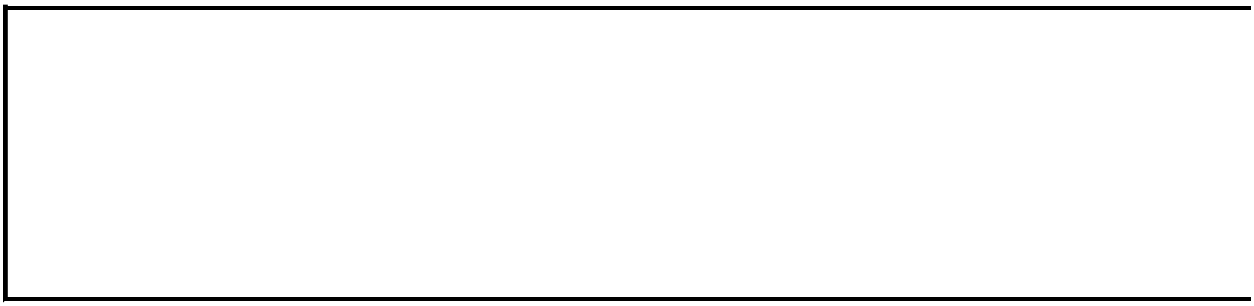
As with all the properties of the **Options** object, the current value of the **AddHebDoubleQuote** property becomes the default setting applied to all new publications.

This property corresponds to the **Add double quotes for Hebrew alphabet numbering** check box on the **Bullets and Numbering** dialog box.

Example

The following example sets Publisher to display double quotes for Hebrew alphabet numbering.

```
Publisher.Options.AddHebDoubleQuote = True
```



Address Property

Returns or sets a **String** that represents the URL address for a hyperlink. Read/write.

expression.**Address**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the URL addresses for all hyperlinks in the active publication.

```
Sub ShowHyperlinkAddresses()
    Dim pgsPage As Page
    Dim shpShape As Shape
    Dim hprLink As Hyperlink
    Dim intCount As Integer
    For Each pgsPage In ActiveDocument.Pages
        For Each shpShape In pgsPage.Shapes
            If shpShape.TextFrame.TextRange.Hyperlinks.Count > 0 The
                For Each hprLink In shpShape.TextFrame.TextRange.Hyp
                    MsgBox "This hyperlink goes to " & hprLink.Addre
                    intCount = intCount + 1
                Next hprLink
            ElseIf shpShape.Hyperlink.Address <> "" Then
                MsgBox "This hyperlink goes to " & shpShape.Hyperlin
                intCount = intCount + 1
            End If
        Next shpShape
    Next pgsPage
    If intCount < 1 Then
        MsgBox "You don't have any hyperlinks in your publication."
    Else
        MsgBox "You have " & intCount & " hyperlinks in " & ThisDocu
    End If
End Sub
```

Adjustments Property

Returns an **Adjustments** collection representing all adjustment handles for the specified **Shape** or **ShapeRange** object.

expression.**Adjustments**

expression Required. An expression that returns one of the objects in the Applies To list.

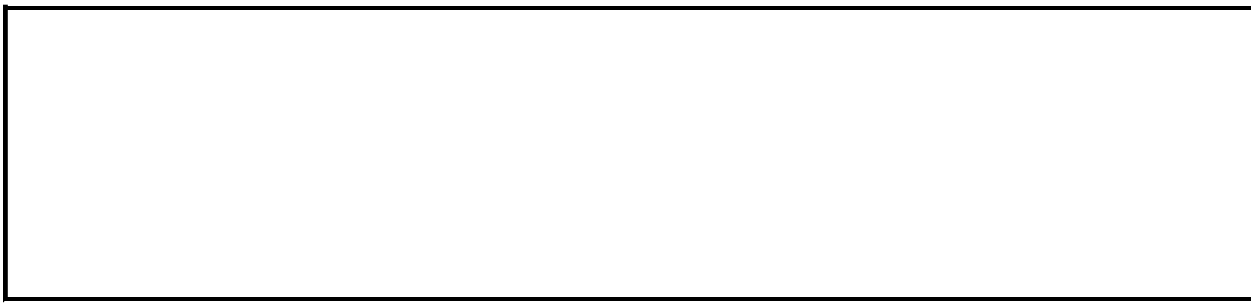
Remarks

Adjustment handles correspond to Microsoft Publisher shape sliders.

Example

This example takes the number of adjustments for a given shape range and assigns it to a variable.

```
Public Sub Counter()

    Dim intCount as Integer

    ' A Shape must be in the active publication and selected.
    intCount = Publisher.ActiveDocument.Selection _
        .ShapeRange(1).Adjustments.Count

End Sub
```



AdvancedPrintOptions Property

Returns an **AdvancedPrintOptions** object that represents the advanced print settings for a publication. Read-only.

expression.**AdvancedPrintOptions**()

expression Required. An expression that returns a **Document** object.

Remarks

The properties of the **AdvancedPrintOptions** object correspond to the options in the **Advanced Print Settings** dialog box.

Example

The following example tests to determine if the active publication has been set to print as separations. If it has, it is set to print only plates for the inks actually used in the publication, and to not print plates for any pages where a color is not used.

```
Sub PrintOnlyInksUsed
        With ActiveDocument.AdvancedPrintOptions
                If .PrintMode = pbPrintModeSeparations Then
                        .InksToPrint = pbInksToPrintUsed
                        .PrintBlankPlates = False
                End If
        End With
End Sub
```


Show All

Alignment Property

 Alignment property as it applies to the **TextEffectFormat** object.

Returns or sets a **MsoTextEffectAlignment** constant that represents the alignment for the specified text effect. Read/write.

MsoTextEffectAlignment can be one of these MsoTextEffectAlignment constants.
msoTextEffectAlignmentCentered
msoTextEffectAlignmentLeft
msoTextEffectAlignmentLetterJustify
msoTextEffectAlignmentMixed
msoTextEffectAlignmentRight
msoTextEffectAlignmentStretchJustify
msoTextEffectAlignmentWordJustify

expression.**Alignment**

expression Required. An expression that returns a **TextEffectFormat** object.

 Alignment property as it applies to the **ParagraphFormat** object.

Returns or sets a **PbParagraphAlignmentType** constant that represents the alignment for the specified paragraphs. Read/write.

PbParagraphAlignmentType can be one of these PbParagraphAlignmentType constants.
pbParagraphAlignmentCenter
pbParagraphAlignmentDistribute
pbParagraphAlignmentDistributeAll
pbParagraphAlignmentDistributeCenterLast
pbParagraphAlignmentDistributeEastAsia
pbParagraphAlignmentInterCluster

pbParagraphAlignmentInterIdeograph
pbParagraphAlignmentInterWord
pbParagraphAlignmentJustified
pbParagraphAlignmentKashida
pbParagraphAlignmentLeft
pbParagraphAlignmentMixed
pbParagraphAlignmentRight

expression.**Alignment**

expression Required. An expression that returns a **ParagraphFormat** object.

 Alignment property as it applies to the **PhoneticGuide** object.

Returns a **PbPhoneticGuideAlignmentType** constant that represents the position of phonetic characters above Japanese text. Read-only.

PbPhoneticGuideAlignmentType can be one of these PbPhoneticGuideAlignmentType constants.
pbPhoneticGuideAlignmentCenter
pbPhoneticGuideAlignmentDefault
pbPhoneticGuideAlignmentLeft
pbPhoneticGuideAlignmentOneTwoOne
pbPhoneticGuideAlignmentRight
pbPhoneticGuideAlignmentZeroOneZero

expression.**Alignment**

expression Required. An expression that returns a **PhoneticGuide** object.

 Alignment property as it applies to the **TabStop** object.

Returns or sets a **PbTabAlignmentType** constant that represents the alignment for the specified tab stop. Read/write.

PbTabAlignmentType can be one of these PbTabAlignmentType constants.

pbTabAlignmentCenter
pbTabAlignmentDecimal
pbTabAlignmentLeading
pbTabAlignmentTrailing

expression.**Alignment**

expression Required. An expression that returns a **TabStop** object.

Example

 As it applies to the **ParagraphFormat** object.

This example adds a new text box to the first page of the active publication, and then add text and sets the paragraph alignment and font formatting.

```
Sub NewTextFrame()
    Dim shpTextBox As Shape
    Set shpTextBox = ActiveDocument.Pages(1).Shapes _
        .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
        Left:=72, Top:=72, Width:=468, Height:=72)
    With shpTextBox.TextFrame.TextRange
        .ParagraphFormat.Alignment = pbParagraphAlignmentCenter
        .Text = "Hello World"
        With .Font
            .Name = "Snap ITC"
            .Size = 30
            .Bold = msoTrue
        End With
    End With
End Sub
```

 As it applies to the **TabStop** object.

This example enters a tabbed list and sets the alignment for two custom tab stops. This example assumes that the specified shape is a text frame and not another type of shape and that there are at least two custom tab stops already set.

```
Sub CustomDecimalTabStop()

    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange
        .InsertAfter Newtext:="Pencils" & vbTab & _
            "Each" & vbTab & "1.50" & vbLf
        .InsertAfter Newtext:="Pens" & vbTab & _
            "Each" & vbTab & "4.95" & vbLf
        .InsertAfter Newtext:="Folders" & vbTab & _
            "Box" & vbTab & "35.28" & vbLf
        .InsertAfter Newtext:="Envelopes" & vbTab & _
            "Case" & vbTab & "150.69" & vbLf
        With .Paragraphs(Start:=1).ParagraphFormat
            .Tabs(1).Alignment = pbTabAlignmentCenter
```

```
            .Tabs(2).Alignment = pbTabAlignmentDecimal
        End With
    End With
End Sub
```


Show All

AllCaps Property

Returns or sets **msoTrue** if the font is formatted as all capital letters or one of the other **MsoTriState** constants if it is not. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse All fonts within the range are not formatted as all caps.

msoTriStateMixed Returned if some fonts in the range are formatted as all caps and others not.

msoTriStateToggle Toggles between **msoTrue** and **msoFalse**.

msoTrue All fonts within the range are formatted with all caps.

expression.**AllCaps**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Setting the **AllCaps** property to **msoTrue** sets the **SmallCaps** property to **msoFalse**, and vice versa.

Example

This example checks the selected text in the active document for text formatted as all capital letters. For this example to work, there must be be an active publication with text selected.

```
Public Sub Caps()

    If Publisher.ActiveDocument.Selection _
        .TextRange.Font.AllCaps = msoTrue Then
        MsgBox "Text is all caps."
    Else
        MsgBox "Text is not all caps."
    End If

End Sub
```

This example formats the selected text as all capital letters. For this code to execute properly, an active document must exist with selected text.

```
Public Sub MakeCaps()

    If Publisher.ActiveDocument.Selection.TextRange _
    .Font.AllCaps = msoFalse Then
            Selection.TextRange.Font.AllCaps = msoTrue
    Else
        MsgBox "You need to select some text" & _

" or it is already all caps."
    End If

End Sub
```

AllowBackgroundSave Property

True (default) for Microsoft Publisher to save publications in the background, allowing users to perform other actions at the same time. Read/write **Boolean**.

expression.**AllowBackgroundSave**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This setting is saved for each individual user and persists from one session to another.

Example

This example turns off background save, so publications do not save in the background.

```
Sub DoNotSaveInBackground()
    Options.AllowBackgroundSave = False
End Sub
```

AllowBleeds Property

True to allow bleeds to print for the specified publication. The default is **True**. Read/write **Boolean**.

expression.**AllowBleeds**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

When bleeds are allowed, objects that are partially off the page print to one eighth inch outside the defined page size.

If you allow bleeds in a document, you can specify whether bleed marks are printed by using the **PrintBleedMarks** property of the **AdvancedPrintOptions** object.

This property corresponds to the **Allow bleeds** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

Example

The following example sets the publication to allow bleeds, and to print bleed marks.

```
Sub AllowBleedsAndPrintMarks()
        With ActiveDocument.AdvancedPrintOptions
                .AllowBleeds = True
                .PrintBleedMarks = True
        End With
End Sub
```

AlternativeText Property

Returns or sets a **String** representing the text displayed by a Web browser in place of the **Shape** object while the **Shape** object is being downloaded or when graphics are turned off. Read/write.

expression.**AlternativeText**

expression Required. An expression that returns one of the objects in the Applies To list.

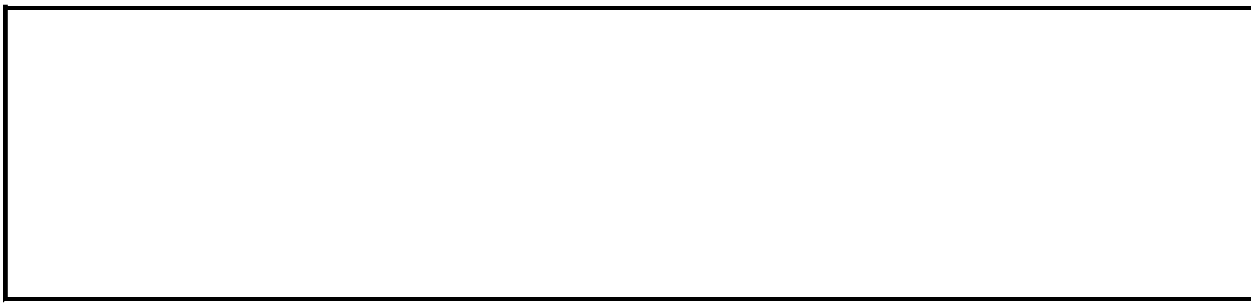
Remarks

The maximum length of the **AlternativeText** property is 254 characters. Microsoft Publisher returns an error if the text length exceeds this number.

Example

This example sets the alternative text for the selected shape in the active document. This example assumes that you have a publication that the selected shape is a picture of a duck.

```
Public Sub Alternative_Text()

    ' The picture of a duck must be selected.
    Publisher.ActiveDocument.Selection.ShapeRange _
        .AlternativeText = "This is a mallard duck."

End Sub
```



AlwaysSaveInDefaultEncoding Property

Returns or sets a **Boolean** value that specifies whether Web pages within a Web publication should always be saved using default encoding. If **True**, Web pages within a publication will always be saved using the default encoding of the client computer. If **False**, Web pages will not be saved using default encoding. The default value is **False**. Read/write.

expression.**AlwaysSaveInDefaultEncoding**

expression Required. An expression that returns a **WebOptions** object.

Remarks

If the **AlwaysSaveInDefaultEncoding** property is set to **True** on a given **WebOptions** object, any subsequent attempts to set the **Encoding** property on that object will be ignored.

Example

The following example tests whether the Web publication is currently set to be saved using default encoding. If so, the **AlwaysSaveInDefaultEncoding** property is set to **False**, and the **Encoding** property is used to set the encoding to Unicode (UTF-8).

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    If .AlwaysSaveInDefaultEncoding = True Then
        .AlwaysSaveInDefaultEncoding = False
        .Encoding = msoEncodingUTF8
    End If
End With
```


Show All

Angle Property

 Angle property as it applies to the **CalloutFormat** object.

Returns or sets an **MsoCalloutAngleType** constant that represents the angle of the callout line. If the callout line contains more than one line segment, this property returns or sets the angle of the segment that is farthest from the callout text box. Read/write.

MsoCalloutAngleType can be one of these MsoCalloutAngleType constants.
msoCalloutAngle30
msoCalloutAngle45
msoCalloutAngle60
msoCalloutAngle90
msoCalloutAngleAutomatic
msoCalloutAngleMixed

expression.**Angle**

expression Required. An expression that returns a **CalloutFormat** object.

Remarks

If you set the value of this property to anything other than **msoCalloutAngleAutomatic**, the callout line maintains a fixed angle as you drag the callout.

 Angle property as it applies to the **PrintablePlate** object.

Returns or sets a **Long** that represents the angle of a printer's color plate. Read/write.

expression.**Angle**

expression Required. An expression that returns one of the above objects.

Remarks

The **InkName** property of the specific **PrintablePlate** object determines its default angle.

InkName:	Default Angle:
pbInkNameBlack	45
pbInkNameCyan	105
pbInkNameMagenta	75
pbInkNameYellow	90
pbInkNameSpot1	45
pbInkNameSpot2	105
pbInkNameSpot3	75
pbInkNameSpot4	30
pbInkNameSpot5	60
pbInkNameSpot6	90
pbInkNameSpot7	135
pbInkNameSpot8	15
pbInkNameSpot9	165
pbInkNameSpot10	120
pbInkNameSpot11	150
pbInkNameSpot12	0

To specify a custom angle setting for a printable plate, the **UseCustomHalftone** of the **AdvancedPrintOptions** object must be set to **True**. Returns "Permission Denied" if the **UseCustomHalftone** is set to **False**.

Example

As it applies to the **CalloutFormat** object.

This example sets the callout angle to 90 degrees for the first shape on the first page of the active publication. For this example to work, the specified shape must be a callout.

```
Sub SetCalloutAngle()
    ActiveDocument.Pages(1).Shapes(1).Callout.Angle = msoCalloutAngl
End Sub
```

As it applies to the **PrintablePlate** object.

This example sets the spot color plates (plates five and higher) of a process and spot color publication to the same custom angle and frequency. The example assumes that the publication's color mode has been specified as process and spot colors, and the publication's print mode has been specified as separations.

```
Sub SetSpotColorPlatesProperties()

ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

Dim intCount As Integer

With ActiveDocument.AdvancedPrintOptions.PrintablePlates
    For intCount = 5 To .Count
        With .Item(intCount)
            .Angle = 45
            .Frequency = 150
        End With
    Next
End With

End Sub
```

Application Property

Used without an object qualifier, this property returns an **Application** object that represents the current instance of Publisher. Used with an object qualifier, this property returns an **Application** object that represents the creator of the specified object. When used with an OLE Automation object, it returns the object's application. Read-only.

expression.**Application**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the version and build information for Publisher.

```
With Application
    MsgBox "Current Publisher: version " _
        & .Version & " build " & .Build
End With
```

This example displays the name of the application that created each linked OLE object on page one of the active publication.

```
Dim shpOle As Shape

For Each shpOle In ActiveDocument.Pages(1).Shapes
    If shpOle.Type = pbLinkedOLEObject Then
        MsgBox shpOle.OLEFormat.Application.Name
    End If
Next
```

Assistant Property

Some of the content in this topic may not be applicable to some languages.

Returns an **Assistant** object that represents the Microsoft Office Assistant.

expression.**Assistant**

expression Required. An expression that returns one of the objects in the Applies To list.

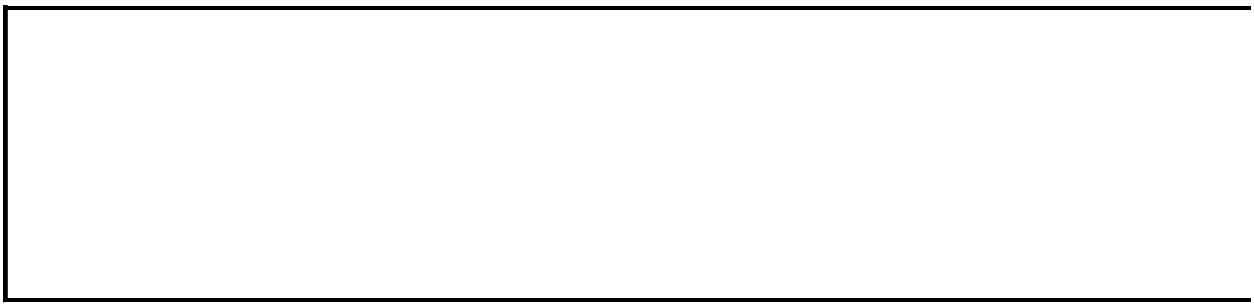
Example

This example displays the Office Assistant.

```
Sub ShowAssistant()
    Assistant.Visible = True
End Sub
```

This example moves the Office Assistant to the upper left corner of the screen and displays a custom message in a balloon.

```
Sub ShowAssistantUpperLeft()
    Dim blnAssistant As Balloon

    With Assistant
        Set blnAssistant = .NewBalloon

        'Moves the Office Assistant
        .Move xLeft:=100, yTop:=100

            'Sets and displays a message with the Office Assistant
            With blnAssistant
                .Mode = msoModeAutoDown
                .Text = "What may I do for you today?"
                .Button = msoButtonSetTipsOptionsClose
                .Show
            End With

    End With

End Sub
```



AttachedToText Property

True if the **Font** or **ParagraphFormat** object is attached to a **TextRange** object. If the object is attached to a **TextRange** object, the document will be updated when properties of the object are changed. If the object is not attached, nothing in the document will be changed until the object is applied to a **TextRange** or **Style** object. Read-only **Boolean**.

expression.**AttachedToText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example duplicates the font formatting; then it checks to see if the duplicated formatting is attached to a text range and if it is not, it attaches the formatting to the second shape.

```
Sub DuplicateText()
    Dim fntTemp As Font
    With ActiveDocument.Pages(1)
        Set fntTemp = .Shapes(1).TextFrame.TextRange.Font.Duplicate
        If fntTemp.AttachedToText <> True Then _
            ActiveDocument.Pages(1).Shapes(2) _
            .TextFrame.TextRange.Font = fntTemp
    End With
End Sub
```


Show All

AutoAttach Property

Returns or sets an **MsoTriState** constant indicating whether the place where the callout line attaches to the callout text box changes depending on whether the origin of the callout line (where the callout points) is to the left or right of the callout text box. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The place where the callout line attaches to the callout text box is fixed.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Set value only; toggles between **msoTrue** and **msoFalse**.

msoTrue The place where the callout line attaches to the callout text box changes depending on the location of the origin of the callout line.

expression.**AutoAttach**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

When the value of this property is **msoTrue**, the drop value (the vertical distance from the edge of the callout text box to the place where the callout line attaches) is measured from the top of the text box when the text box is to the right of the origin, and it's measured from the bottom of the text box when the text box is to the left of the origin. When the value of this property is **msoFalse**, the drop value is always measured from the top of the text box, regardless of the relative positions of the text box and the origin. Use the **CustomDrop** method to set the drop value, and use the **Drop** property to return the drop value.

Setting this property affects a callout only if it has an explicitly set drop value — that is, if the value of the **DropType** property is **msoCalloutDropCustom**. By default, callouts have explicitly set drop values when they're created.

Example

This example adds two callouts to the first page. One of the callouts is automatically attached and the other is not. If you change the callout line origin for the automatically attached callout to the right of the attached text box, the position of the text box changes. The callout that is not automatically attached does not display this behavior.

```
With ActivePublication.Pages(1).Shapes
    With .AddCallout(Type:=msoCalloutTwo, _
            Left:=420, Top:=170, Width:=200, Height:=50)
        .TextFrame.TextRange.Text = "auto-attached"
        .Callout.AutoAttach = msoTrue
    End With
    With .AddCallout(Type:=msoCalloutTwo, _
            Left:=420, Top:=350, Width:=200, Height:=50)
        .TextFrame.TextRange.Text = "not auto-attached"
        .Callout.AutoAttach = msoFalse
    End With
End With
```


Show All

AutoFitText Property

Sets or returns a **PbTextAutoFitType** constant that represents how Microsoft Publisher automatically adjusts the text font size and the **TextFrame** objects size for best viewing. Read/write.

PbTextAutoFitType can be one of these PbTextAutoFitType constants.

pbTextAutoFitBestFit Text frame size adjusts to fit text.

pbTextAutoFitNone Allows text to overflow the text frame.

pbTextAutoFitShrinkOnOverflow Text font reduces so text fits within the text frame.

expression.**AutoFitText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example tests to see if the text frame has text, and if so, the **AutoFitText** property is set to best fit.

```
Sub TextFit()

    Dim tfFrame As TextFrame

    tfFrame = Application.ActiveDocument.MasterPages.Item(1).Shapes(
    With tfFrame
        If .HasText = msoTrue Then .AutoFitText = pbTextAutoFitBestF
    End With

End Sub
```

AutoFormatWord Property

True for Microsoft Publisher to automatically format the entire word where the insertion point exists, even when no text is selected. Read/write **Boolean**.

expression.**AutoFormatWord**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If only one or two characters in a word is selected, only the selected characters will be affected by a formatting change not the whole word.

Example

This example sets global options for Microsoft Publisher, including enabling automatic formatting of the entire word.

```
Sub SetGlobalOptions()
    With Options
        .AutoFormatWord = True
        .AutoKeyboardSwitching = True
        .AutoSelectWord = True
        .DragAndDropText = True
        .UseCatalogAtStartup = False
        .UseHelpfulMousePointers = False
    End With
End Sub
```

AutoHyphenate Property

True (default) for Microsoft Publisher to automatically hyphenate text in text frames. Read/write **Boolean**.

expression.**AutoHyphenate**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example turns on automatic hyphenation for Publisher and sets the amount of space from the right margin to use when hyphenating words to one inch (72 points).

```
Sub SetHyphenationZone()
    With Options
        .AutoHyphenate = True
        .HyphenationZone = 72
    End With
End Sub
```

AutoKeyboardSwitching Property

True for Microsoft Publisher to automatically switch the keyboard language to the language used for the text at the insertion point. Read/write **Boolean**.

expression.**AutoKeyboardSwitching**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enables automatically switching the keyboard language to the necessary language.

```
Sub SetGlobalOptions()
    Options.AutoKeyboardSwitching = True
End Sub
```


Show All

AutoLength Property

Returns an **MsoTriState** constant indicating whether the first segment of the callout line is scaled when the callout is moved. Applies only to callouts whose lines consist of more than one segment (types **msoCalloutThree** and **msoCalloutFour**). Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The first segment of the callout retains the fixed length specified by the **Length** property whenever the callout is moved.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The first segment of the callout line (the segment attached to the text callout box) is scaled automatically whenever the callout is moved.

expression.**AutoLength**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **AutomaticLength** method to set this property to **mso True**, and use the **CustomLength** method to set this property to **msoFalse**.

Example

This example toggles between an automatically-scaling first segment and one with a fixed length for the callout line for the first shape in the publication. For the example to work, the shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    If .AutoLength Then
        .CustomLength Length:=50
    Else
        .AutomaticLength
    End If
End With
```

AutomaticPairKerningThreshold Property

Returns or sets a **Variant** value that represents the point size above which kerning is automatically adjusted for characters in the specified text range. Read/write.

expression.**AutomaticPairKerningThreshold**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Valid range is 0.0 points to 999.5 points. Returns –2 if the value for characters in the text range is indeterminate. Setting this property to 0.0 disables automatic pair kerning on the range.

Example

This example sets the point size threshold to 12 points. All text in the second story above the threshold will implement auto kerning.

```
Sub Threshold()

    Application.ActiveDocument.Stories(2).TextRange _
        .Font.AutomaticPairKerningThreshold = 12

End Sub
```

AutoSelectWord Property

True for Microsoft Publisher to automatically select the entire word when selecting text. Read/write **Boolean**.

expression.**AutoSelectWord**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets Microsoft Publisher global options, including enabling automatically selecting an entire word when selecting text.

```
Sub SetGlobalOptions()
    With Options
        .AutoFormatWord = True
        .AutoKeyboardSwitching = True
        .AutoSelectWord = True
        .DragAndDropText = True
        .UseCatalogAtStartup = False
        .UseHelpfulMousePointers = False
    End With
End Sub
```


Show All

AutoShapeType Property

Returns or sets an **MsoAutoShapeType** which specifies a **Shape** object's AutoShape type.

MsoAutoShapeType can be one of these MsoAutoShapeType constants.

msoShapeMoon
msoShape16pointStar
msoShape24pointStar
msoShape32pointStar
msoShape4pointStar
msoShape5pointStar
msoShape8pointStar
msoShapeActionButtonBackorPrevious
msoShapeActionButtonBeginning
msoShapeActionButtonCustom
msoShapeActionButtonDocument
msoShapeActionButtonEnd
msoShapeActionButtonForwardorNext
msoShapeActionButtonHelp
msoShapeActionButtonHome
msoShapeActionButtonInformation
msoShapeActionButtonMovie
msoShapeActionButtonReturn
msoShapeActionButtonSound
msoShapeArc
msoShapeBalloon
msoShapeBentArrow
msoShapeBentUpArrow
msoShapeBevel
msoShapeBlockArc
msoShapeCan

msoShapeChevron
msoShapeCircularArrow
msoShapeCloudCallout
msoShapeCross
msoShapeCube
msoShapeCurvedDownArrow
msoShapeCurvedDownRibbon
msoShapeCurvedLeftArrow
msoShapeCurvedRightArrow
msoShapeCurvedUpArrow
msoShapeCurvedUpRibbon
msoShapeDiamond
msoShapeDonut
msoShapeDoubleBrace
msoShapeDoubleBracket
msoShapeDoubleWave
msoShapeDownArrow
msoShapeDownArrowCallout
msoShapeDownRibbon
msoShapeExplosion1
msoShapeExplosion2
msoShapeFlowchartAlternateProcess
msoShapeFlowchartCard
msoShapeFlowchartCollate
msoShapeFlowchartConnector
msoShapeFlowchartData
msoShapeFlowchartDecision
msoShapeFlowchartDelay
msoShapeFlowchartDirectAccessStorage
msoShapeFlowchartDisplay
msoShapeFlowchartDocument
msoShapeFlowchartExtract
msoShapeFlowchartInternalStorage

msoShapeFlowchartMagneticDisk
msoShapeFlowchartManualInput
msoShapeFlowchartManualOperation
msoShapeFlowchartMerge
msoShapeFlowchartMultidocument
msoShapeFlowchartOffpageConnector
msoShapeFlowchartOr
msoShapeFlowchartPredefinedProcess
msoShapeFlowchartPreparation
msoShapeFlowchartProcess
msoShapeFlowchartPunchedTape
msoShapeFlowchartSequentialAccessStorage
msoShapeFlowchartSort
msoShapeFlowchartStoredData
msoShapeFlowchartSummingJunction
msoShapeFlowchartTerminator
msoShapeFoldedCorner
msoShapeHeart
msoShapeHexagon
msoShapeHorizontalScroll
msoShapeIsoscelesTriangle
msoShapeLeftArrow
msoShapeLeftArrowCallout
msoShapeLeftBrace
msoShapeLeftBracket
msoShapeLeftRightArrow
msoShapeLeftRightArrowCallout
msoShapeLeftRightUpArrow
msoShapeLeftUpArrow
msoShapeLightningBolt
msoShapeLineCallout1
msoShapeLineCallout1AccentBar
msoShapeLineCallout1BorderandAccentBar

msoShapeLineCallout1NoBorder
msoShapeLineCallout2
msoShapeLineCallout2AccentBar
msoShapeLineCallout2BorderandAccentBar
msoShapeLineCallout2NoBorder
msoShapeLineCallout3
msoShapeLineCallout3AccentBar
msoShapeLineCallout3BorderandAccentBar
msoShapeLineCallout3NoBorder
msoShapeLineCallout4
msoShapeLineCallout4AccentBar
msoShapeLineCallout4BorderandAccentBar
msoShapeLineCallout4NoBorder
msoShapeMixed
msoShapeNoSymbol
msoShapeNotchedRightArrow
msoShapeNotPrimitive
msoShapeOctagon
msoShapeOval
msoShapeOvalCallout
msoShapeParallelogram
msoShapePentagon
msoShapePlaque
msoShapeQuadArrow
msoShapeQuadArrowCallout
msoShapeRectangle
msoShapeRectangularCallout
msoShapeRegularPentagon
msoShapeRightArrow
msoShapeRightArrowCallout
msoShapeRightBrace
msoShapeRightBracket
msoShapeRightTriangle

msoShapeRoundedRectangle
msoShapeRoundedRectangularCallout
msoShapeSmileyFace
msoShapeStripedRightArrow
msoShapeSun
msoShapeTrapezoid
msoShapeUpArrow
msoShapeUpArrowCallout
msoShapeUpDownArrow
msoShapeUpDownArrowCallout
msoShapeUpRibbon
msoShapeUTurnArrow
msoShapeVerticalScroll
msoShapeWave

expression.**AutoShapeType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

AutoShapes correspond to **Shape** objects, although the **AutoShapeType** property for non-Publisher shapes will also return a value. WordArt, OLE, Web Form control, table and picture frame objects should return **msoShapeMixed** as their **AutoShapeType** property value. Text frames should return **msoShapeRectangle** as their **AutoShapeType** property.

Example

This example converts the selected **AutoShape** object to a lightning bolt if it is a heart and to a 5-point star if it is not. For this example to execute properly, you must have an **AutoShape** object selected in the active publication.

```
Sub ShapeShift()

    Dim srShift As ShapeRange

    Set srShift = Application.ActiveDocument.Selection.ShapeRange
    If srShift.AutoShapeType = msoShapeHeart Then
        srShift.AutoShapeType = msoShapeLightningBolt
    Else
        srShift.AutoShapeType = msoShape5pointStar
    End If

End Sub
```

AutoUpdate Property

True if all pages will be added to the specified Web navigation bar set and that adding new pages will update the navigation bar with a corresponding item. Pages must have the **AddHyperlinkToWebNavbar** set to **True** or **WebPageOptions.IncludePageOnNewWebNavigationBars** property set to **True** to be added or updated within the specified **WebNavigationBarSet**. Read/write **Boolean**.

expression.**AutoUpdate**()

expression Required. An expression that returns a **WebNavigationBarSet** object.

Remarks

This property determines whether or not the existing pages in the publication will be added to the navigation bar and if added pages will also be updated. These pages must be marked with the **AddHyperlinkToWebNavbar** set to **True** or **WebPageOptions.IncludePageOnNewWebNavigationBars** property set to **True** to be added or updated within the specified **WebNavigationBarSet**. Changing this setting does not change the number of items in the bar, it just determines whether or not new pages will be added. By setting this value to **False** it is possible to design specific navigation bars for specific content pages in a Web site that do not contain all of the available hyperlinks in the publication.

The default value is **True**.

Example

The following example adds a new Web navigation bar set to the active document with text style buttons and auto update set to **False** so that page links will not be added or new pages automatically updated in the navigation bar, then the Web navigation bar is added to the first page of the publication.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets.AddSet(Name:="ne
With objWebNav
    .AutoUpdate = False
    .ButtonStyle = pbnbButtonStyleText
End With
ActiveDocument.Pages(1).Shapes.AddWebNavigationBar _
    Name:="newBar", Left:=10, Top:=10
```

AvailableLabels Property

Returns the collection of **Label** objects that represent each unique label design available on the system. Read-only.

expression.**AvailableLabels**

expression Required. An expression that returns a **PageSetup** object.

Remarks

The **Labels** collection contains the members returned by the **AvailableLabels** property.

Members of the **AvailableLabels** collection are identical to the list of labels available from the Page Setup dialog box.

Example

The following example returns the fifth label available on the system by using **AvailableLabels**(*index*), and then some of the label's properties are set.

```
Dim theLabel As Label

With ActiveDocument.PageSetup
    .Label = .AvailableLabels(5)  ' Label 5 is Avery 5164
    Set theLabel = .Label
    With theLabel
        .LeftMargin = InchesToPoints(0.15)
        .TopMargin = InchesToPoints(0.15)
        .HorizontalGap = InchesToPoints(0.1)
        .VerticalGap = InchesToPoints(0.1)
    End With
End With
```

BackColor Property

Returns or sets a **ColorFormat** object representing the background color for the specified fill or patterned line. Read/write.

expression.**BackColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **ForeColor** property to set the foreground color for a fill or line.

Example

This example adds a rectangle to the active publication and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
With ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeRectangle, _
        Left:=90, Top:=90, Width:=90, Height:=50).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(170, 170, 170)
    .TwoColorGradient _
        Style:=msoGradientHorizontal, Variant:=1
End With
```

This example adds a patterned line to the active publication.

```
With ActiveDocument.Pages(1).Shapes.AddLine _
        (BeginX:=10, BeginY:=100, EndX:=250, EndY:=0).Line
    .Weight = 6
    .ForeColor.RGB = RGB(0, 0, 255)
    .BackColor.RGB = RGB(128, 0, 0)
    .Pattern = msoPatternDarkDownwardDiagonal
End With
```

Background Property

Sets or returns a **PageBackground** object representing the background of the specified page.

expression.**Background**

expression Required. An expression that returns a **Page** object.

Remarks

This proerty is for publication pages only. Any attempt to create a background for a master page will return a "Permission denied" error.

Example

The following example creates a **PageBackground** object and sets it to the background of the first page of the active document.

```
Dim objPageBackground As PageBackground
Set objPageBackground = ActiveDocument.Pages(1).Background
```

BackgroundSound Property

Returns or sets a **String** that specifies the path to a sound file that is played when the Web page is loaded in a Web browser. Read/write.

expression.**BackgroundSound**

expression Required. An expression that returns a **WebPageOptions** object.

Remarks

The path to the background sound file must be a network path or a local path; an http:// address will not work.

If **BackgroundSound** is specified, the background sound will play once by default. The **SetBackgroundSoundRepeat** method can be used to specify whether the background sound should be played infinitely, and if it should not, to specify the number of times the background sound file should be played.

The background sound can be any of the following file types.

*.wav
*.mid
*.midi
*.rmi
*.au
*.aif
*.aiff

Example

The following example sets the background sound for page four of the active Web publication to a .wav file on the local computer. This .wav file will play once when the page is loaded in a Web browser.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(4).WebPageOptions

With theWPO
    .BackgroundSound = "C:\CompanySounds\corporate_jingle.wav"
End With
```

BackgroundSoundLoopCount Property

Returns a **Long** value that specifies the number of times the background sound attached to a Web page is played when the page is loaded in a Web browser. Read-only.

expression.**BackgroundSoundLoopCount**

expression Required. An expression that returns a **WebPageOptions** object.

Remarks

The **SetBackgroundSoundRepeat** method can be used to specify the number of times the background sound file is played when the page is loaded. If using the **SetBackgroundSoundRepeat** method to specify the number of times the background file is played, the **BackgroundSoundLoopCount** property will be equal to that specified value. Note that valid values range from 1 to 999, inclusive. Attempting to set this value outside this range will result in a run-time error.

Until the **SetBackgroundSoundRepeat** method is used to change the number of times the background sound file is played, **BackgroundSoundLoopCount** is 1.

Example

The following example sets the background sound for page four of the active Web publication to a .wav file on the local computer. If **BackgroundSoundLoopCount** is less than three, the **SetBackgroundSoundRepeat** method is used to specify that the background sound be repeated three times. The **BackgroundSoundLoopCount** property will now be three.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(4).WebPageOptions

With theWPO
    .BackgroundSound = "C:\CompanySounds\corporate_jingle.wav"
    If .BackgroundSoundLoopCount < 3 Then
        .SetBackgroundSoundRepeat RepeatForever:=False, RepeatTimes:
    End If
End With
```

BackgroundSoundLoopForever Property

Returns a **Boolean** value that specifies whether the background sound attached to the Web page should be repeated infinitely. Read-only.

expression.**BackgroundSoundLoopForever**

expression Required. An expression that returns a **WebPageOptions** object.

Remarks

The **SetBackgroundSoundRepeat** method is used to specify whether the background sound should be repeated infinitely after the page is loaded. Until the **SetBackgroundSoundRepeat** method is used to specify whether the background sound should be played infinitely, **BackgroundSoundLoopForever** is **False**.

Example

The following example sets the background sound for page four of the active Web publication to a .wav file on the local computer. If **BackgroundSoundLoopForever** is **False**, the **SetBackgroundSoundRepeat** method is used to specify that the background sound should be repeated infinitely. The **BackgroundSoundLoopForever** property will now be **True**.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(4).WebPageOptions

With theWPO
    .BackgroundSound = "C:\CompanySounds\corporate_jingle.wav"
    If .BackgroundSoundLoopForever = False Then
        .SetBackgroundSoundRepeat RepeatForever:=True
    End If
End With
```


Show All

BaseRGB Property

Returns or sets an **MsoRGBType** constant that represents the original RGB color format before color-changing properties, such as tinting and shading, are applied. Read/write.

expression.**BaseRGB**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a shape, sets the fill color and lightens the color; then it creates a second shape and applies the original RGB color of the first shape to the second shape.

```
Sub SetBaseRGB()
    Dim shpOne As Shape

    With ActiveDocument.Pages(1).Shapes
        Set shpOne = .AddShape(Type:=msoShapeHeart, _
            Left:=150, Top:=150, Width:=300, Height:=300)
        With shpOne.Fill.ForeColor
            .RGB = RGB(Red:=160, Green:=0, Blue:=255)
            .TintAndShade = 0.9
        End With
        .AddShape(Type:=msoShapeRectangle, Left:=62, _
            Top:=500, Width:=488, Height:=100).Fill _
            .ForeColor.RGB = shpOne.Fill.ForeColor.BaseRGB
    End With
End Sub
```

BaseStyle Property

Returns or sets a **String** that represents the style upon which the formatting of another style is based. Read/write.

expression.**BaseStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the base formatting of the style named Body Text to the formatting of the Normal style.

```
Sub SetBaseStyle()
    With ActiveDocument.TextStyles
        .Add "Body Text"
        .Item("Body Text").BaseStyle = "Normal"
    End With
End Sub
```

BaseText Property

Returns a **String** that represents the text to which the specified phonetic text applies. Read-only.

expression.**BaseText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds phonetic text to the selection and displays the text to which the phonetic text applies, which is the originally-selected text. This example assumes text is selected. If no text is selected, the message box will be blank.

```
Sub AddPhoneticText()
    With Selection.TextRange.Fields.AddPhoneticGuide _
            (Range:=Selection.TextRange, Text:="tray sheek")
        MsgBox "The base text is " & .PhoneticGuide.BaseText
    End With
End Sub
```


Show All

BeginArrowheadLength Property

Returns or sets an **MsoArrowheadLength** constant indicating the length of the arrowhead at the beginning of the specified line. Read/write.

MsoArrowheadLength can be one of these MsoArrowheadLength constants.

msoArrowheadLengthMedium

msoArrowheadLengthMixed Return value only; indicates a combination of the other states in the specified shape range.

msoArrowheadLong

msoArrowheadShort

expression.**BeginArrowheadLength**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **EndArrowheadLength** property to return or set the length of the arrowhead at the end of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

BeginArrowheadStyle Property

Returns or sets an **MsoArrowheadStyle** constant indicating the style of the arrowhead at the beginning of the specified line. Read/write.

MsoArrowheadStyle can be one of these MsoArrowheadStyle constants.

msoArrowheadDiamond

msoArrowheadNone

msoArrowheadOpen

msoArrowheadOval

msoArrowheadStealth

msoArrowheadStyleMixed Return value only; indicates a combination of the other states in the specified shape range.

msoArrowheadTriangle

expression.**BeginArrowheadStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **EndArrowheadStyle** property to return or set the style of the arrowhead at the end of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

BeginArrowheadWidth Property

Returns or sets an **MsoArrowheadWidth** constant indicating the width of the arrowhead at the beginning of the specified line. Read/write.

MsoArrowheadWidth can be one of these MsoArrowheadWidth constants.

msoArrowheadNarrow

msoArrowheadWide

msoArrowheadWidthMedium

msoArrowheadWidthMixed Return value only; indicates a combination of the other states in the specified shape range.

expression.**BeginArrowheadWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **EndArrowheadWidth** property to return or set the width of the arrowhead at the end of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

BeginConnected Property

Returns an **MsoTriState** constant indicating whether the beginning of the specified connector is connected to a shape. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The beginning of the specified connector is not connected to a shape.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The beginning of the specified connector is connected to a shape.

expression.**BeginConnected**

expression Required. An expression that returns one of the objects in the Applies To list.

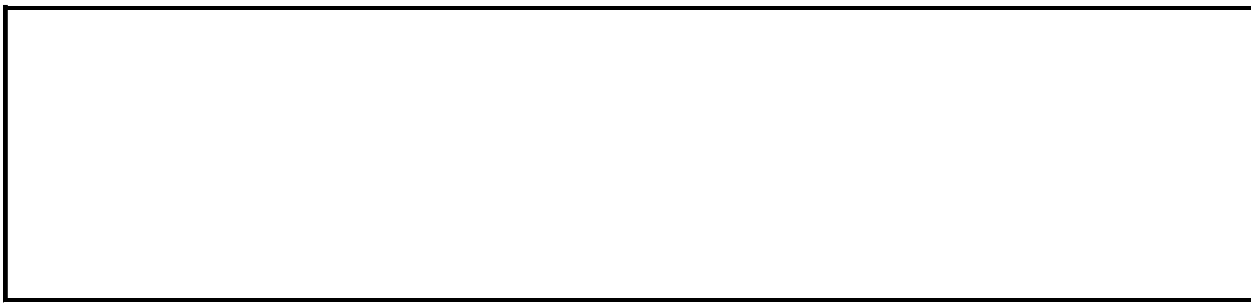
Remarks

Use the **EndConnected** property to determine if the end of a connector is connected to a shape.

Example

If the third shape on the first page in the active publication is a connector whose beginning is connected to a shape, this example stores the connection site number, stores a reference to the connected shape, and then disconnects the beginning of the connector from the shape.

```
Dim intSite As Integer
Dim shpConnected As Shape

With ActiveDocument.Pages(1).Shapes(3)

    ' Test whether shape is a connector.
    If .Connector Then
        With .ConnectorFormat

            ' Test whether connector is connected to another shape.
            If .BeginConnected Then

                ' Store connection site number.
                intSite = .BeginConnectionSite

                ' Set reference to connected shape.
                Set shpConnected = .BeginConnectedShape

                ' Disconnect connector and shape.
                .BeginDisconnect
            End If
        End With
    End If
End With
```



BeginConnectedShape Property

Returns a **Shape** object that represents the shape to which the beginning of the specified connector is attached.

expression.**BeginConnectedShape**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the beginning of the specified connector isn't attached to a shape, an error occurs.

Use the **EndConnectedShape** property to return the shape attached to the end of a connector.

Example

This example assumes that the first page in the active publication already contains two shapes attached by a connector named Conn1To2. The code adds a rectangle and a connector to the first page. The beginning of the new connector will be attached to the same connection site as the beginning of the connector named Conn1To2, and the end of the new connector will be attached to connection site one on the new rectangle.

```
Dim shpNew As Shape
Dim intSite As Integer
Dim shpOld As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add new rectangle.
    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=450, Top:=190, Width:=200, Height:=100)

    ' Add new connector.
    .AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=10, EndY:=10) _
        .Name = "Conn1To3"

    ' Get connection site number of old shape, and set
    ' reference to old shape.
    With .Item("Conn1To2").ConnectorFormat
        intSite = .BeginConnectionSite
        Set shpOld = .BeginConnectedShape
    End With

    ' Connect new connector to old shape and new rectangle.
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect ConnectedShape:=shpOld, _
            ConnectionSite:=intSite
        .EndConnect ConnectedShape:=shpNew, _
            ConnectionSite:=1
    End With
End With
```

BeginConnectionSite Property

Returns a **Long** indicating the connection site to which the beginning of a connector is connected. Read-only.

expression.**BeginConnectionSite**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the beginning of the specified connector isn't attached to a shape, this property generates an error.

Use the **EndConnectionSite** property to return the site to which the end of a connector is connected.

Example

This example assumes that the first page in the active publication already contains two shapes attached by a connector named Conn1To2. The code adds a rectangle and a connector to the first page. The beginning of the new connector will be attached to the same connection site as the beginning of the connector named Conn1To2, and the end of the new connector will be attached to connection site one on the new rectangle.

```
Dim shpNew As Shape
Dim intSite As Integer
Dim shpOld As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add new rectangle.
    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=450, Top:=190, Width:=200, Height:=100)

    ' Add new connector.
    .AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=10, EndY:=10) _
        .Name = "Conn1To3"

    ' Get connection site number of old shape, and set
    ' reference to old shape.
    With .Item("Conn1To2").ConnectorFormat
        intSite = .BeginConnectionSite
        Set shpOld = .BeginConnectedShape
    End With

    ' Connect new connector to old shape and new rectangle.
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect ConnectedShape:=shpOld, _
            ConnectionSite:=intSite
        .EndConnect ConnectedShape:=shpNew, _
            ConnectionSite:=1
    End With
End With
```


Show All

Black Property

Sets or returns a **Long** that represents the black component of a CMYK color. Value can be any number between 0 and 255. Read/write.

expression.**Black**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two new shapes and then sets the CMYK fill color for one shape and sets the CMYK values of the second shape to the same CMYK values.

```
Sub ReturnAndSetCMYK()
    Dim lngCyan As Long
    Dim lngMagenta As Long
    Dim lngYellow As Long
    Dim lngBlack As Long
    Dim shpHeart As Shape
    Dim shpStar As Shape

    Set shpHeart = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeHeart, Left:=100, _
        Top:=100, Width:=100, Height:=100)
    Set shpStar = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=100, Width:=150, Height:=150)

    With shpHeart.Fill.ForeColor.CMYK
       .SetCMYK 10, 80, 200, 30
        lngCyan = .Cyan
        lngMagenta = .Magenta
        lngYellow = .Yellow
        lngBlack = .Black
    End With

    'Sets new shape to current shape's CMYK colors
    shpStar.Fill.ForeColor.CMYK.SetCMYK _
        Cyan:=lngCyan, Magenta:=lngMagenta, _
        Yellow:=lngYellow, Black:=lngBlack
End Sub
```


Show All

BlackWhiteMode Property

Returns or sets an **MsoBlackWhiteMode** constant indicating how the specified shape or shape range appears when the publication is viewed in black-and-white mode. Read/write.

MsoBlackWhiteMode can be one of these MsoBlackWhiteMode constants.

msoBlackWhiteAutomatic

msoBlackWhiteBlack

msoBlackWhiteBlackTextAndLine

msoBlackWhiteDontShow

msoBlackWhiteGrayOutline

msoBlackWhiteGrayScale

msoBlackWhiteHighContrast

msoBlackWhiteInverseGrayScale

msoBlackWhiteLightGrayScale

msoBlackWhiteMixed Return value only; indicates a combination of the other states in the specified shape range.

msoBlackWhiteWhite

expression.**BlackWhiteMode**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the first shape in the active publication to appear in black-and-white mode. When you view the publication in black-and-white mode, the shape will appear black, regardless of what color it is in color mode.

```
ActiveDocument.Pages(1).Shapes(1) _
    .BlackWhiteMode = msoBlackWhiteBlack
```


Show All

Bold Property

Returns or sets an **MsoTriState** property that represents the state of the **Bold** property on the characters in a text range. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters in the range are formatted as bold.

msoTriStateMixed Return value indicating that the range contains some text formatted as bold and some text not formatted as bold.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All characters in the range are formatted as bold.

expression.**Bold**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests all the text in the second story of the active publication and if it has mixed bolding, it sets all the text to bold. If the text is all bold or all not bold, a message is displayed informing the user there is no mixed bolding. For this code to execute properly, there need to be two or more stories with text in the active publication.

```
Sub BoldStory()

    Dim stf As Publisher.Font

    Set stf = Application.ActiveDocument.Stories(2).TextRange.Font
    With stf
        If .Bold = msoTriStateMixed Then
            .Bold = msoTrue
        Else
            MsgBox "Mixed bolding is not in this story."
        End If
    End With

End Sub
```


Show All

BoldBi Property

Returns or sets an **MsoTriState** constant indicating if the font is bold; used with text in a right-to-left language. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters in the range are formatted as bold.

msoTriStateMixed Return value indicating that the range contains some text formatted as bold and some text not formatted as bold.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All characters in the range are formatted as bold.

expression.**BoldBi**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the text in the first story and displays one of two possible messages depending on if the text is right-to-left formatted and if its font is bold. For this example to execute properly, there must be at least one story with text in the active publication.

```
Sub BoldRtoL()

    Dim stf As Font

    Set stf = Application.ActiveDocument.Stories(1).TextRange.Font

    With stf
        If .BoldBi = msoTrue Then
            MsgBox "This story is right-to-left and is bold."
        Else
            MsgBox "This story is either not right-to-left" & _
                " or it is not bold."
        End If
    End With

End Sub
```


Show All

Border Property

Returns or sets an **MsoTriState** constant indicating whether the text in the specified callout is surrounded by a border. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The text in the specified callout is not surrounded by a border.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Set value only; toggles between **msoTrue** and **msoFalse**.

msoTrue The text in the specified callout is surrounded by a border.

expression.**Border**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active publication and a callout that points to the oval. The callout text will have a border, but not a vertical accent bar that separates the text from the callout line.

```
With ActiveDocument.Pages(1).Shapes
    ' Add an oval.
    .AddShape Type:=msoShapeOval, _
        Left:=180, Top:=200, Width:=280, Height:=130

    ' Add a callout.
    With .AddCallout(Type:=msoCalloutTwo, _
            Left:=420, Top:=170, Width:=170, Height:=40)

        ' Add text to the callout.
        .TextFrame.TextRange.Text = "This is an oval"

        ' Add an accent bar to the callout.
        With .Callout
            .Accent = msoFalse
            .Border = msoTrue
        End With
    End With
End With
```

BorderArt Property

Returns a **BorderArtFormat** object that represents the BorderArt type applied to the specified shape. Returns "Permission Denied" if BorderArt has not been applied to the shape. Read-only.

expression.**BorderArt**()

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

BorderArt are picture borders that can be applied to text boxes, picture frames, or rectangles.

Use the **BorderArt** property to apply, change, and remove BorderArt from shapes in publications.

Example

The following example tests for the existence of BorderArt on each shape for each page of the active publication. If BorderArt exists, it is deleted.

```
Sub DeleteBorderArt()

Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Delete
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

BorderArts Property

Returns a **BorderArts** collection that represents the BorderArt types available for use in the specified publication. Read-only.

expression.**BorderArts**()

expression Required. An expression that returns a **Document** object.

Remarks

BorderArt are picture borders that can be applied to text boxes, picture frames, or rectangles.

The **BorderArts** collection includes any custom BorderArt types created by the user for the specified publication.

Example

The following example returns the BorderArts collection and lists the names of all the BorderArt types available for use in the active publication.

```
Sub ListBorderArt()
Dim bdaTemp As BorderArts
Dim bdaLoop As BorderArt

Set bdaTemp = ActiveDocument.BorderArts

For Each bdaLoop In bdaTemp
    Debug.Print "The name of this BorderArt is " & bdaLoop.Name
Next bdaLoop
End Sub
```

BorderBottom Property

Returns a **CellBorder** object that represents the bottom border for a specified table cell.

expression.**BorderBottom**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a checkerboard design using borders and a fill color with an existing table. This assumes the first shape on page two is a table and not another type of shape and that the table has an uneven number of columns.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell
    Dim intCell As Integer

    intCell = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            With celTable
                With .BorderBottom
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderTop
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderLeft
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderRight
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
            End With
            If intCell Mod 2 = 0 Then
               celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
            intCell = intCell + 1
        Next celTable
    Next rowTable
```

```
End Sub
```

BorderDiagonal Property

Returns a **CellBorder** object that represents the diagonal border for a specified table cell.

expression.**BorderDiagonal**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example diagonally splits every other cell in the specified table and adds a diagonal border. This example assumes the first shape on page two is a table and not another type of shape.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell
    Dim intCell As Integer

    intCell = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            If intCell Mod 2 = 0 Then
                With celTable
                    .Diagonal = pbTableCellDiagonalDown
                    With .BorderDiagonal
                        .Weight = 1
                        .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                    End With
                End With
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
            intCell = intCell + 1
        Next celTable
    Next rowTable

End Sub
```

BorderLeft Property

Returns a **CellBorder** object that represents the left border for a specified table cell.

expression.**BorderLeft**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a checkerboard design using borders and a fill color with an existing table. This assumes the first shape on page two is a table and not another type of shape and that the table has an uneven number of columns.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell
    Dim intCell As Integer

    intCell = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            With celTable
                With .BorderBottom
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderTop
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderLeft
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderRight
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
            End With
            If intCell Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
            intCell = intCell + 1
        Next celTable
    Next rowTable
```

```
End Sub
```

BorderRight Property

Returns a **CellBorder** object that represents the right border for a specified table cell.

expression.**BorderRight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a checkerboard design using borders and a fill color with an existing table. This assumes the first shape on page two is a table and not another type of shape and that the table has an uneven number of columns.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell
    Dim intCell As Integer

    intCell = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            With celTable
                With .BorderBottom
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderTop
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderLeft
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderRight
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
            End With
            If intCell Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
            intCell = intCell + 1
        Next celTable
    Next rowTable
```

```
End Sub
```

BorderTop Property

Returns a **CellBorder** object that represents the top border for a specified table cell.

expression.**BorderTop**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a checkerboard design using borders and a fill color with an existing table. This assumes the first shape on page two is a table and not another type of shape and that the table has an uneven number of columns.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell
    Dim intCell As Integer

    intCell = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            With celTable
                With .BorderBottom
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderTop
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderLeft
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
                With .BorderRight
                    .Weight = 2
                    .Color.RGB = RGB(Red:=0, Green:=0, Blue:=0)
                End With
            End With
            If intCell Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
            intCell = intCell + 1
        Next celTable
    Next rowTable
```

```
End Sub
```

BoundHeight Property

Returns a **Single** indicating the height, in points, of the bounding box for the specified text range. Read-only.

expression.**BoundHeight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the position, width, and height of the bounding box surrounding the text in the first shape on page one of the active publication.

```
Dim rngText As TextRange
Dim strMessage As String

Set rngText = ActiveDocument.Pages(1) _
    .Shapes(1).TextFrame.TextRange

With rngText
    strMessage = "Text frame information" & vbCrLf _
        & "    Distance from left edge of page: " _
        & .BoundLeft & " points" & vbCrLf _
        & "    Distance from top edge of page: " _
        & .BoundTop & " points" & vbCrLf _
        & "    Width: " & .BoundWidth & " points" & vbCrLf _
        & "    Height: " & .BoundHeight & " points"
End With

MsgBox strMessage
```

BoundLeft Property

Returns a **Single** indicating the distance, in points, from the left edge of the leftmost page to the left edge of the bounding box for the specified text range. Read-only.

expression.**BoundLeft**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the position, width, and height of the bounding box surrounding the text in the first shape on page one of the active publication.

```
Dim rngText As TextRange
Dim strMessage As String

Set rngText = ActiveDocument.Pages(1) _
    .Shapes(1).TextFrame.TextRange

With rngText
    strMessage = "Text frame information" & vbCrLf _
        & "    Distance from left edge of page: " _
        & .BoundLeft & " points" & vbCrLf _
        & "    Distance from top edge of page: " _
        & .BoundTop & " points" & vbCrLf _
        & "    Width: " & .BoundWidth & " points" & vbCrLf _
        & "    Height: " & .BoundHeight & " points"
End With

MsgBox strMessage
```

BoundTop Property

Returns a **Single** indicating the distance, in points, from the top edge of the topmost page to the top edge of the bounding box for the specified text range. Read-only.

expression.**BoundTop**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the position, width, and height of the bounding box surrounding the text in the first shape on page one of the active publication.

```
Dim rngText As TextRange
Dim strMessage As String

Set rngText = ActiveDocument.Pages(1) _
    .Shapes(1).TextFrame.TextRange

With rngText
    strMessage = "Text frame information" & vbCrLf _
        & "    Distance from left edge of page: " _
        & .BoundLeft & " points" & vbCrLf _
        & "    Distance from top edge of page: " _
        & .BoundTop & " points" & vbCrLf _
        & "    Width: " & .BoundWidth & " points" & vbCrLf _
        & "    Height: " & .BoundHeight & " points"
End With

MsgBox strMessage
```

BoundWidth Property

Returns a **Single** indicating the width, in points, of the bounding box for the specified text range. Read-only.

expression.**BoundWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the position, width, and height of the bounding box surrounding the text in the first shape on page one of the active publication.

```
Dim rngText As TextRange
Dim strMessage As String

Set rngText = ActiveDocument.Pages(1) _
    .Shapes(1).TextFrame.TextRange

With rngText
    strMessage = "Text frame information" & vbCrLf _
        & "    Distance from left edge of page: " _
        & .BoundLeft & " points" & vbCrLf _
        & "    Distance from top edge of page: " _
        & .BoundTop & " points" & vbCrLf _
        & "    Width: " & .BoundWidth & " points" & vbCrLf _
        & "    Height: " & .BoundHeight & " points"
End With

MsgBox strMessage
```

Brightness Property

Returns or sets a **Single** indicating the brightness of the specified picture or OLE object. The value for this property must be a number from 0.0 (dimmest) to 1.0 (brightest). Read/write.

expression.**Brightness**

expression Required. An expression that returns one of the objects in the Applies To list.

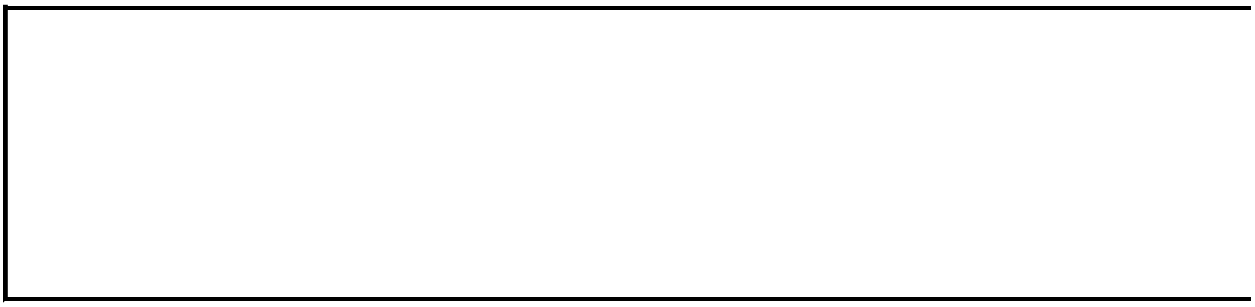
Remarks

Use the **IncrementBrightness** method to incrementally adjust the brightness from its current level.

Example

This example sets the brightness for the first shape in the active publication. The shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(1).PictureFormat _
    .Brightness = 0.3
```



Build Property

Returns a **Long** that represents the Microsoft Publisher build number. Read-only.

expression.**Build**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the the build number of Publisher.

```
Sub BuildNumber()
    MsgBox Prompt:="The current Microsoft Publisher build number is
        Application.Build, Title:="Microsoft Publisher Build"
End Sub
```


Show All

ButtonStyle Property

Sets or returns a **PbWizardNavBarButtonStyle** constant that represents the style of the navigation bar buttons: large, small, or text-only. Read/write.

pbButtonStyleLarge
pbButtonStyleSmall
pbButtonStyleText

expression.**ButtonStyle**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Example

The following example sets the button style to **pbnbButtonStyleLarge** for the first Web navigation bar set of the active document.

```
ActiveDocument.WebNavigationBarSets(1).ButtonStyle = pbnbButtonStyle
```

ButtonText Property

Returns or sets a **String** that represents the text that appears on the face of a Web command button. Read/write.

expression.**ButtonText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web command button, assigns text to appear on its face, and specifies an e-mail address to which to send the form data.

```
Sub NewWebForm()
    With ActiveDocument.Pages.Add(Count:=1, After:=1)
        With .Shapes.AddWebControl(Type:=pbWebControlCommandButton,
                Left:=72, Top:=72, Width:=72, Height:=36)
            With .WebCommandButton
                .ButtonType = pbCommandButtonSubmit
                .ButtonText = "Send Form:"
                .EmailAddress = "someone@microsoft.com"
            End With
        End With
    End With
End Sub
```


Show All

ButtonType Property

Returns or sets a **PbCommandButtonType** constant that indicates whether a Web command button will clear or submit form data. Read/write.

PbCommandButtonType can be one of these PbCommandButtonType constants.

pbCommandButtonReset

pbCommandButtonSubmit

expression.**ButtonType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web command submit button, assigns text to appear on its face, and specifies an e-mail address to which to send the form data.

```
Sub NewWebForm()
    With ActiveDocument.Pages.Add(Count:=1, After:=1)
        With .Shapes.AddWebControl(Type:=pbWebControlCommandButton,
                Left:=72, Top:=72, Width:=72, Height:=36)
            With .WebCommandButton
                .ButtonType = pbCommandButtonSubmit
                .ButtonText = "Send Form:"
                .EmailAddress = "someone@example.com"
            End With
        End With
    End With
End Sub
```

Callout Property

Returns a **CalloutFormat** object representing the formatting of a line callout.

expression.**Callout**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active publication and a callout that points to the oval. The callout text won't have a border, but it will have a vertical accent bar that separates the text from the callout line.

```
Sub NewShapeItem()

    Dim shpNew As Shapes

    Set shpNew = Application.ActiveDocument.MasterPages(1).Shapes
    With shpNew
        .AddShape Type:=msoShapeOval, Left:=180, _
            Top:=200, Width:=280, Height:=130
        With .AddCallout(Type:=msoCalloutTwo, Left:=420, _
            Top:=170, Width:=170, Height:=40)
            .TextFrame.TextRange = "Big Oval"
            With .Callout
                .Accent = msoTrue
                .Border = msoFalse
            End With
        End With
    End With

End Sub
```

Caption Property

Returns or sets a **String** indicating the caption at the top of the Microsoft Publisher application window. Read/write.

expression.**Caption**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example demonstrates how a subroutine could temporarily change the Publisher window caption and then restore it afterwards.

```
Sub WindowCaption()
    Dim strCaption As String

    strCaption = ActiveWindow.Caption

    ActiveWindow.Caption = "Custom process--please wait..."

    ' Run custom code here.

    ActiveWindow.Caption = strCaption
End Sub
```


Show All

CatalogMergeItems Property

Returns a **CatalogMergeShapes** collection that represents the shapes included in the catalog merge area. Read-only.

expression.**CatalogMergeItems**

expression Required. An expression that returns a **Shape** object.

Remarks

The catalog merge area can contain picture and text data fields you have inserted, as well as other design elements you choose.

Example

The following example tests whether any page in the specified publication contains a catalog merge area, and if it does it returns a list of the shapes it contains.

```
Sub ListCatalogMergeAreaContents()

    Dim pgPage As Page
    Dim mmLoop As Shape
    Dim intCount As Integer

    For Each pgPage In ThisDocument.Pages
        For Each mmLoop In pgPage.Shapes

            If mmLoop.Type = pbCatalogMergeArea Then

                With mmLoop.CatalogMergeItems
                    For intCount = 1 To .Count
                        Debug.Print "Shape ID: " & _
                            mmLoop.CatalogMergeItems.Item(intCount).
                        Debug.Print "Shape Name: " & _
                            mmLoop.CatalogMergeItems.Item(intCount).
                    Next
                End With

            End If

        Next mmLoop
    Next pgPage

End Sub
```


Show All

Cells Property

 Cells property as it applies to the **Column** and **Row** objects.

Returns a **CellRange** object that represents the cell or cells in a column or row of a table.

expression.**Cells**

expression Required. An expression that returns one of the above objects.

 Cells property as it applies to the **Table** object.

Returns a **CellRange** object that represents a range of cells in a table.

expression.**Cells**(***StartRow***, ***StartColumn***, ***EndRow***, ***EndColumn***)

expression Required. An expression that returns a **Table** object.

StartRow Optional **Long**. The row in which the starting cell exists. If this argument is omitted, all the table rows are included in the range.

StartColumn Optional **Long**. The column in which the starting cell exists. If this argument is omitted, all the table columns are included in the range.

EndRow Optional **Long**. The row in which the ending cell exists. If this argument is omitted, only the row specified by ***StartRow*** is included in the range. If this argument is specified but ***StartRow*** is omitted, an error occurs.

EndColumn Optional **Long**. The column in which the ending cell exists. If this argument is omitted, only the column specified by ***StartColumn*** is included in the range. If this argument is specified but ***StartColumn*** is omitted, an error occurs.

Remarks

If all arguments are omitted, all the cells in the table are included in the range.

Example

 As it applies to the **Column** and **Row** objects.

This example merges the first and second cells in the first column of the specified table.

```
Sub MergeCell()
    With ActiveDocument.Pages(1).Shapes(2).Table.Columns(1)
        .Cells(1).Merge MergeTo:=.Cells(2)
    End With
End Sub
```

This example applies a thick border outline to the first cell in the second column of the specified table.

```
Sub OutlineBorderCell()
    With ActiveDocument.Pages(1).Shapes(2).Table.Columns(2).Cells(1)
        .BorderLeft.Weight = 5
        .BorderRight.Weight = 5
        .BorderTop.Weight = 5
        .BorderBottom.Weight = 5
    End With
End Sub
```

 As it applies to the **Table** object.

This example merges the first two cells in the first two rows of the specified table.

```
Sub MergeCells()
    ActiveDocument.Pages(1).Shapes(2).Table _
        .Cells(StartRow:=1, StartColumn:=1, _
        EndRow:=2, EndColumn:=2).Merge
End Sub
```


Show All

CellTextOrientation Property

Returns or sets a **PbTextOrientation** that represents the flow of text in a specified table cell. Read/write.

PbTextOrientation can be one of these PbTextOrientation constants.

pbTextOrientationHorizontal

pbTextOrientationMixed

pbTextOrientationRightToLeft

pbTextOrientationVerticalEastAsia

expression.**CellTextOrientation**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example increases the height of the cells in the first row, and then adds vertically-oriented heading text.

```
Sub VerticalText()
    Dim rowTable As Row
    Dim celTable As Cell

    With ActiveDocument.Pages(2).Shapes(1).Table.Rows(1)
        .Height = Application.InchesToPoints(1.5)
        For Each celTable In .Cells
            With celTable
                .CellTextOrientation _
                    = pbTextOrientationVerticalEastAsia
                .TextRange.ParagraphFormat.Alignment _
                    = pbParagraphAlignmentCenter
                .TextRange.Text = "Column Heading " _
                    & celTable.Column
            End With
        Next
    End With
End Sub
```

CharBasedFirstLineIndent Property

Returns or sets the value of the first line indent (in East Asian character width). Read/write **Long**.

expression.**CharBasedFirstLineIndent**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

The value of **CharBasedFirstLineIndent** can be returned or set only after the **UseCharBasedFirstLineIndent** has been set. A run-time "permission denied" error is returned if **UseCharBasedFirstLineIndent** is not set first. Note, however, that **UseCharBasedFirstLineIndent** can be set only if East Asian languages are enabled on the client computer (the value can be returned regardless of whether East Asian languages are enabled). This effectively means that **CharBasedFirstLineIndent** cannot be used unless East Asian languages are enabled on the client computer.

The value of **CharBasedFirstLineIndent** can range from 0 (zero) to 80.

Example

The following example creates a text box on the fourth page of the active publication. After the **UseCharBasedFirstLineIndent** property is set to **True**, the width of the first line indent is set to 15 points by using the **CharBasedFirstLineIndent** property. Font properties are then set, and text is inserted into the paragraph.

```
Dim theTextBox As Shape

Set theTextBox = ActiveDocument.Pages(4).Shapes _
        .AddShape(msoShapeRectangle, 100, 100, 300, 200)

With theTextBox
    .TextFrame.TextRange.ParagraphFormat _
        .UseCharBasedFirstLineIndent = msoTrue
    .TextFrame.TextRange.ParagraphFormat _
        .CharBasedFirstLineIndent = 15
    .TextFrame.TextRange.Font.Name = "Verdana"
    .TextFrame.TextRange.Font.Size = 12
    .TextFrame.TextRange.Text = "This is a test sentence." _
       & Chr(13) & "This is another test sentence."
End With
```

ChildShapeRange Property

Returns a **ShapeRange** object representing the child shapes of a selection.

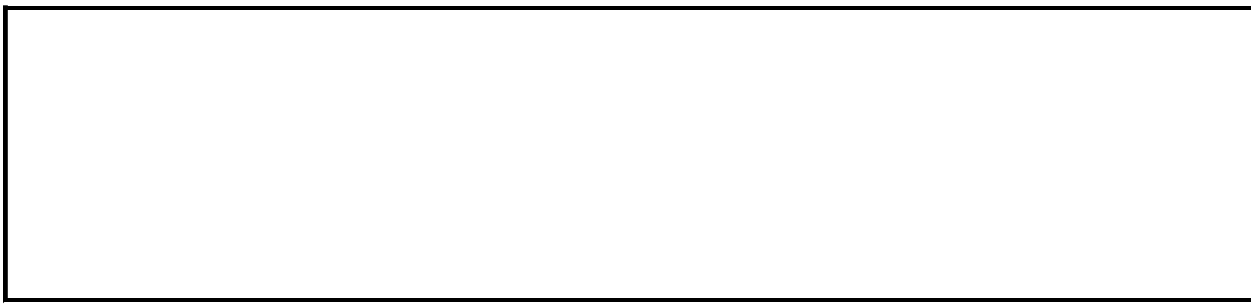
expression.**ChildShapeRange**

expression Required. An expression that returns a **Selection** object.

Example

This example creates a new page in the active publication, populates the page with shapes, and selects and groups the shapes. Then after deselecting two of the group shapes, it changes the AutoShape type for one of the shapes.

```
Sub ChangeFillToChildShape()

    With ThisDocument.Pages(1)
        With .Shapes
            .AddShape msoShape4pointStar, 10, 10, 175, 175
            .AddShape msoShapeOval, 100, 100, 175, 75
            .AddShape msoShapeOval, 150, 150, 175, 75
            .Range.Group
            .SelectAll
        End With
        .Shapes(1).GroupItems(1).Select msoFalse
        .Shapes(1).GroupItems(2).Select msoFalse
    End With

    Selection.ChildShapeRange(3).AutoShapeType = msoShapeDiamond

End Sub
```




Show All

CMYK Property

Returns a **ColorCMYK** object that represents CMYK color properties.

expression.**CMYK**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two new shapes and then sets the CMYK fill color for one shape and sets the CMYK values of the second shape to the same CMYK values.

```
Sub ReturnAndSetCMYK()
    Dim lngCyan As Long
    Dim lngMagenta As Long
    Dim lngYellow As Long
    Dim lngBlack As Long
    Dim shpHeart As Shape
    Dim shpStar As Shape

    Set shpHeart = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeHeart, Left:=100, _
        Top:=100, Width:=100, Height:=100)
    Set shpStar = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=100, Width:=150, Height:=150)

    With shpHeart.Fill.ForeColor.CMYK
       .SetCMYK 10, 80, 200, 30
        lngCyan = .Cyan
        lngMagenta = .Magenta
        lngYellow = .Yellow
        lngBlack = .Black
    End With

    'Sets new shape to current shape's CMYK colors
    shpStar.Fill.ForeColor.CMYK.SetCMYK _
        Cyan:=lngCyan, Magenta:=lngMagenta, _
        Yellow:=lngYellow, Black:=lngBlack
End Sub
```

Code Property

Returns a **String** that represents the text displayed when the page view is set to show field codes. Read-only.

expression.**Code**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example loops through all the fields in the active publication, and then displays a message as to whether the string "www" was found in the code of any of the fields.

```
Sub FindWWWHyperlinks()
    Dim intItem As Integer
    Dim intField As Integer
    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange.Field
        Do
            intItem = intItem + 1
            If InStr(1, .Item(intItem).Code, "www") > 0 Then
                intField = intField + 1
            End If
        Loop Until intItem = .Count
    End With

    If intField > 0 Then
        MsgBox "You have " & intField & " World Wide Web " & _
            "hyperlinks in your publication."
    Else
        MsgBox "You have no hyperlink fields in your publication."
    End If
End Sub
```

Color Property

Returns a **ColorFormat** object representing the color information for the specified object.

expression.**Color**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the font color of the first story in the active document and tells the user if the font color is black or not.

```
Sub FontColor()

    If Application.ActiveDocument.Stories(1) _
        .TextRange.Font.Color.RGB = RGB(Red:=0, Green:=0, Blue:=0) T
        MsgBox "Your font color is black"
    Else
        MsgBox "Your font color is not black"
    End If

End Sub
```


Show All

ColorMode Property

Returns a **PbColorMode** constant that represents the color mode for the publication. Read-only.

PbColorMode can be one of these PbColorMode constants.

pbColorModeBW

pbColorModeDesktop

pbColorModeProcess

pbColorModeSpot

pbColorModeSpotAndProcess

expression.**ColorMode**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a spot-color plate collection, adds two plates to it, and then enters those plates into the spot-color mode.

```
Sub CreateSpotColorMode()
    Dim plArray As Plates

    With ThisDocument
        'Creates a color plate collection,
        'which contains one black plate by default
        Set plArray = .CreatePlateCollection(Mode:=pbColorModeSpot)

        'Sets the plate color to red
        plArray(1).Color.RGB = RGB(255, 0, 0)

        'Adds another plate, black by default and
        'sets the plate color to green
        plArray.Add
        plArray(2).Color.RGB = RGB(0, 255, 0)

        'Enters spot color mode with above
        'two plates in the plates array
        If .ColorMode = pbColorModeSpot Then
            .EnterColorMode pbColorModeSpot, plArray
        End If
    End With
End Sub
```


Show All

ColorModel Property

Returns a **PbColorModel** constant that represents the color model of the picture. Read-only.

PbColorModel can be one of these PbColorModel constants.
PbColorModelCMYK
PbColorModelGreyScale
PbColorModelRGB
PbColorModelUnknown

expression.**ColorModel**()

expression Required. An expression that returns a **PictureFormat** object.

Example

The following example returns a list of the pictures with RGB color mode in the active publication.

```
Sub ListRGBPictures()
Dim pgLoop As Page
Dim shpLoop As Shape

    For Each pgLoop In ActiveDocument.Pages
        For Each shpLoop In pgLoop.Shapes

            If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPi

                With shpLoop.PictureFormat
                    If .IsEmpty = msoFalse Then
                        If .ColorModel = pbColorModelRGB Then
                            Debug.Print .Filename
                        End If
                    End If
                End With

            End If

        Next shpLoop
    Next pgLoop

End Sub
```


Show All

Colors Property

Returns a **ColorFormat** object representing a color from the specified color scheme.

expression.**Colors**(***ColorIndex***)

expression Required. An expression that returns one of the objects in the Applies To list.

ColorIndex Required **PbSchemeColorIndex**. The color from the scheme to return based on its function in the scheme.

PbSchemeColorIndex can be one of these PbSchemeColorIndex constants.

pbSchemeColorAccent1
pbSchemeColorAccent2
pbSchemeColorAccent3
pbSchemeColorAccent4
pbSchemeColorAccent5
pbSchemeColorFollowedHyperlink
pbSchemeColorHyperlink
pbSchemeColorMain
pbSchemeColorNone

Example

The following example loops through the **ColorSchemes** collection and looks for color schemes where the followed hyperlink color matches the color with the RGB value of 128.

```
Dim cscLoop As ColorScheme
Dim colTemp As ColorFormat

For Each cscLoop In Application.ColorSchemes
    With cscLoop
        Set colTemp = .Colors(ColorIndex:=pbSchemeColorFollowedHyper
        If colTemp.RGB = RGB(128, 0, 0) Then
            Debug.Print "Color scheme '" & .Name _
                & "' has a followed hyperlink " _
                & "color matching RGB(128, 0, 0)"
        End If
    End With
Next cscLoop
```

ColorScheme Property

Returns or sets the **ColorScheme** object that represents the scheme colors for the specified publication. Read/write.

expression.**ColorScheme**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the name of the current color scheme for the active publication.

```
With ActiveDocument.ColorScheme
    MsgBox "The current color scheme is " & .Name & "."
End With
```

This example sets the color scheme of the active publication to "Alpine."

```
ActiveDocument.ColorScheme _
    = Application.ColorSchemes("Alpine")
```

ColorSchemes Property

Returns a **ColorSchemes** collection that represents the color schemes available.

expression.**ColorSchemes**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example loops through the **ColorSchemes** collection and displays the name of each color scheme and the RGB value of the color for followed hyperlinks in each scheme.

```
Dim cscLoop As ColorScheme
Dim cscAll As ColorSchemes

Set cscAll = Application.ColorSchemes

For Each cscLoop In cscAll
    With cscLoop
        Debug.Print "Color scheme: " & .Name _
            & " / Followed hyperlink color: " _
            & .Colors(ColorIndex:=pbSchemeColorFollowedHyperlink).RG
    End With
Next cscLoop
```

ColorsInPalette Property

Returns a **Long** that represents the number of colors in the picture's palette. Read-only.

expression.**ColorsInPalette**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

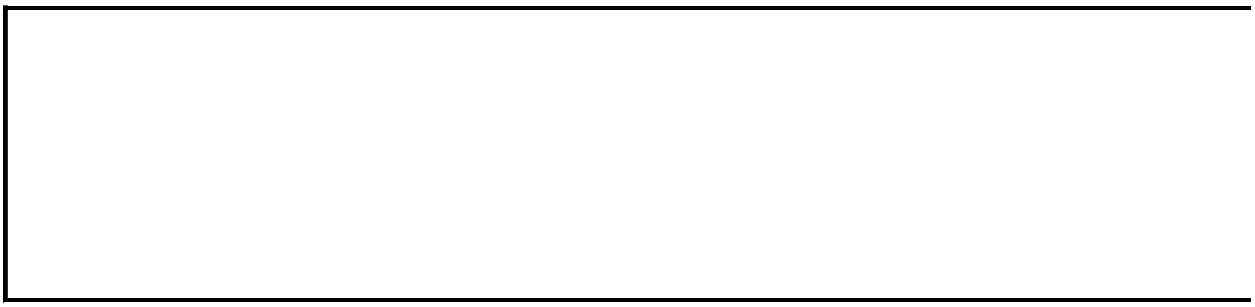
This property only applies to pictures that are not TrueColor (that is, pictures that contain color data of less than 24 bits per channel.) Returns "Permission Denied" for shapes representing pictures that are TrueColor.

Use the **IsTrueColor** property of the **PictureFormat** object to determine whether a picture contains color data of 24 bits per channel or greater.

Example

The following example tests each picture in the active document, and prints out whether the picture is TrueColor, and if not, how many colors are in the picture's palette.

```
For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Or shpLoop.Type = pbPictur

            With shpLoop.PictureFormat
                If .IsEmpty = msoFalse Then
                    Debug.Print .Filename
                    If .IsTrueColor = msoTrue Then
                        Debug.Print "This picture is TrueColor"
                    Else
                        Debug.Print "This picture contains " & .Colo
                    End If
                End If
            End With

        End If
    Next shpLoop
Next pgLoop
```




Show All

ColorsInUse Property

Returns a **ColorsInUse** collection that represents the colors present in the current publication. Read-only.

expression.**ColorsInUse**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **ColorsInUse** collection supports all the publication color models: RGB, process colors, and spot color.

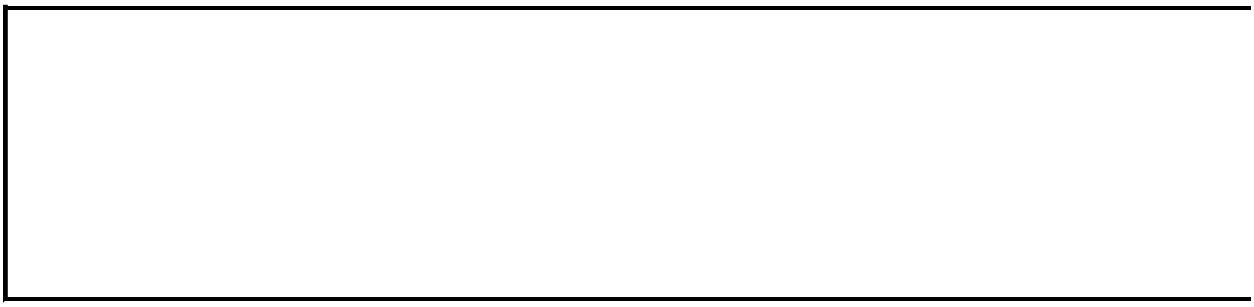
For process color and spot color publications, colors are based on inks. For a given ink, a publication may contain several colors that are different tints or shades of that ink. Use the **Plates** collection to access the plates that represent the inks defined for a publication.

This property corresponds to the **Colors** tab of the **Color Printing** dialog box.

Example

The following example lists properties of each color in the active publication that is based on the specified ink. This example assumes the publication's color mode has been defined as spot color or process and spot color.

```
Sub ListColorsBasedOnInk()
Dim cfLoop As ColorFormat

For Each cfLoop In ActiveDocument.ColorsInUse

    With cfLoop
        If .Ink = "2" Then
            Debug.Print "BaseRGB: " & .BaseRGB
            Debug.Print "RGB: " & .RGB
            Debug.Print "TintShade: " & .TintAndShade
            Debug.Print "Type: " & .Type
        End If
    End With

Next cfLoop

End Sub
```




Show All

ColorType Property

Returns or sets an **MsoPictureColorType** constant indicating the type of color transformation applied to the specified picture or OLE object. Read/write.

MsoPictureColorType can be one of these MsoPictureColorType constants.

msoPictureAutomatic

msoPictureBlackAndWhite

msoPictureGrayscale

msoPictureMixed Return value only; indicates a combination of the other states in the specified shape range.

msoPictureWatermark

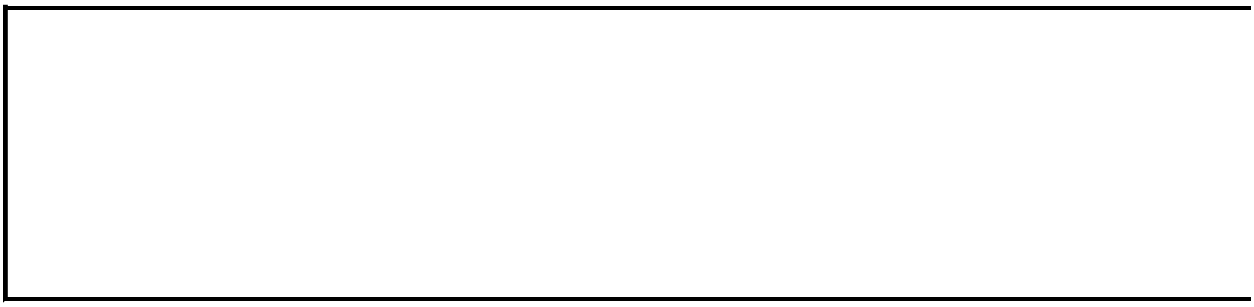
expression.**ColorType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the color transformation to grayscale for the first shape in the active publication. The shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(1).PictureFormat _
    .ColorType = msoPictureGrayScale
```




Show All

Column Property

 Column property as it applies to the **Cell** and **CellRange** objects.

Returns a **Long** that represents the table column containing the specified cell. Read-only.

expression.**Column**

expression Required. An expression that returns one of the above objects.

 Column property as it applies to the **MailMergeFilterCriterion** object.

Returns a **String** that represents the name of the field in the mail merge data source to use in the filter. Read/write.

expression.**Column**

expression Required. An expression that returns one of the above objects.

Example

Example as it applies to the **Cell** and **CellRange** objects.

This example adds a page to the active publication, creates a table on that new page, and diagonally splits all cells in even-numbered columns.

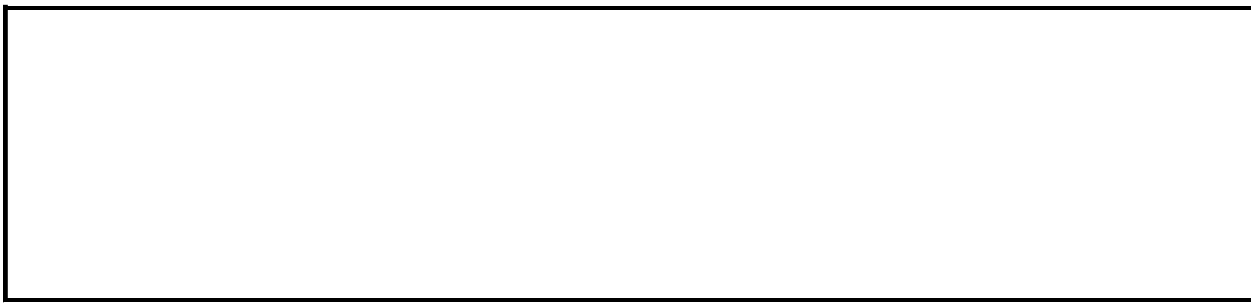
```
Sub CreateNewTable()

    Dim pgeNew As Page
    Dim shpTable As Shape
    Dim tblNew As Table
    Dim celTable As Cell
    Dim rowTable As Row

    'Creates a new document with a five-row by five-column table
    Set pgeNew = ActiveDocument.Pages.Add(Count:=1, After:=1)
    Set shpTable = pgeNew.Shapes.AddTable(NumRows:=5, NumColumns:=5,
        Left:=72, Top:=72, Width:=468, Height:=100)
    Set tblNew = shpTable.Table

    'Inserts a diagonal split into all cells in even-numbered column
    For Each rowTable In tblNew.Rows
        For Each celTable In rowTable.Cells
            If celTable.Column Mod 2 = 0 Then
                celTable.Diagonal = pbTableCellDiagonalUp
            End If
        Next celTable
    Next rowTable

End Sub
```

Example as it applies to the **MailMergeFilterCriterion** object.

The following example changes an existing filter to remove from the mail merge all records that do not have a Region field equal to "WA".

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
```

```
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next intItem
    End With
End Sub
```



ColumnGutterWidth Property

Returns or sets the width of the column gutters that are used by the **LayoutGuides** object to aid in the process of laying out design elements. Read/write **Single**.

expression.**ColumnGutterWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The default width of column gutters is 0.4 inches.

Example

The following example modifies the second master page of the active publication so that it has four rows and four columns, row gutter width of 0.75 inches, column gutter width of 0.5 inches, and center lines in the gutters. Any new pages added to the publication that use the second master page as a template will have these properties.

```
Dim theMasterPage As page
Dim theLayoutGuides As LayoutGuides

Set theMasterPage = ActiveDocument.MasterPages(2)
Set theLayoutGuides = theMasterPage.LayoutGuides

With theLayoutGuides
    .Rows = 4
    .Columns = 4
    .RowGutterWidth = Application.InchesToPoints(0.75)
    .ColumnGutterWidth = Application.InchesToPoints(0.5)
    .GutterCenterlines = True
End With
```


Show All

Columns Property

 Columns property as it applies to the **LayoutGuides** and **TextFrame** objects.

Returns or sets a **Long** that represents the number of guide columns on a page or the number of columns in a text frame. Read/write.

expression.**Columns**

expression Required. An expression that returns one of the above objects.

 Columns property as it applies to the **Table** object.

Returns a **Columns** collection that represents all the columns of the specified table.

***expression*.Columns**

expression Required. An expression that returns one of the above objects.

Example

 As it applies to the **LayoutGuide** objects.

This example adds a new page with a text box and formats the active publication with two guide columns and the new text box with two newspaper-type columns.

```
Sub LayoutTwoColumnPage()
    Dim shpTextBox As Shape
    With ActiveDocument
        .Pages.Add Count:=1, After:=1
        Set shpTextBox = .Pages(2).Shapes.AddTextbox _
            (Orientation:=pbTextOrientationHorizontal, _
            Left:=72, Top:=72, Width:=468, Height:=318)
        With .LayoutGuides
            .Columns = 2
            .Rows = 2
        End With
        With shpTextBox.TextFrame
            .Columns = 2
        End With
    End With
End Sub
```

 As it applies to the **Table** object.

This example enters a bold number into each cell in the specified table. This example assumes the specified shape is a table and not another type of shape.

```
Sub CountCellsByColumn()
    Dim shpTable As Shape
    Dim colTable As Column
    Dim celTable As Cell
    Dim intCount As Integer

    intCount = 1

    Set shpTable = ActiveDocument.Pages(2).Shapes(1)

    'Loops through each column in the table
    For Each colTable In shpTable.Table.Columns

        'Loops through each cell in the column
```

```
        For Each celTable In colTable.Cells
            With celTable.Text
                .Text = intCount
                .ParagraphFormat.Alignment = _
                    pbParagraphAlignmentCenter
                .Font.Bold = msoTrue
                intCount = intCount + 1
            End With
        Next celTable
    Next colTable

End Sub
```

ColumnSpacing Property

Returns or sets a **Variant** that represents the amount of space between text columns. Read/write.

expression.**ColumnSpacing**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Spacing measures from the end of the text to the end of the column and again from the beginning of the column to the beginning of the text. Thus, if you enter a **ColumnSpacing** amount of 0.5, the total spacing between columns will be one inch: half an inch measuring from the end of the text to the end of the column in one column, and half an inch measuring from the beginning of the column to the beginning of the text in a neighboring column.

Example

This example formats the first text box in the active publication with three columns and a total of half an inch spacing between columns.

```
Sub SetColumnsAndSpacing()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame
        .Columns = 3
        .ColumnSpacing = InchesToPoints(0.25)
    End With
End Sub
```


Show All

COMAddIns Property

Returns a **COMAddIns** collection that represents a reference to the Component Object Model (COM) add-ins currently loaded in Publisher.

expression.**COMAddIns**

expression Required. An expression that returns an **Application** object.

Remarks

These are listed in the **COM Add-Ins** dialog box. You can add the **Add-Ins** command to your **Tools** menu by using the **Customize** dialog box.

CommandBars Property

Sets or returns a **CommandBars** collection that represents the menu bar and all the toolbars in Microsoft Publisher.

expression.**CommandBars**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enlarges all command bar buttons, enables ToolTips, and shows all menu items when displaying menus.

```
Sub CmdBars()

    With CommandBars
        .LargeButtons = False
        .DisplayTooltips = True
        .AdaptiveMenus = False
    End With

End Sub
```

This example displays the **Objects** toolbar at the bottom of the application window.

```
Sub ShowObjectsToolbar

    With CommandBars("Objects")
        .Visible = True
        .Position = msoBarBottom
    End With

End Sub
```

CompareTo Property

Returns or sets a **String** that represents the text to compare in the query filter criterion. Read/write.

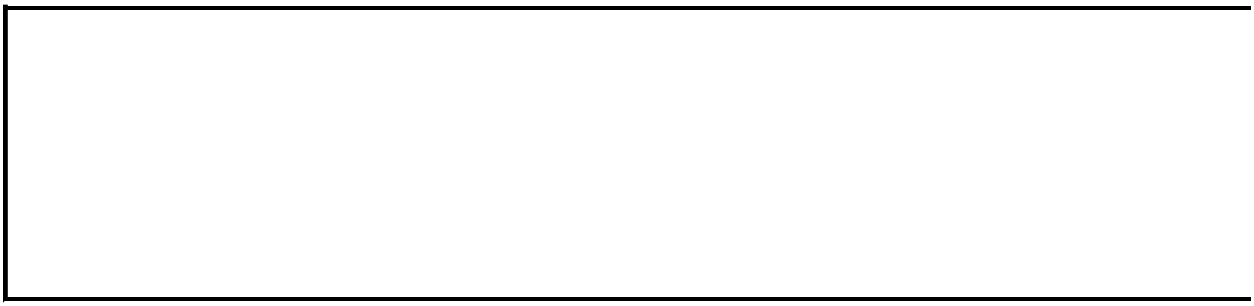
expression.**CompareTo**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example changes an existing filter to remove from the mail merge all records that do not have a Region field equal to "WA". This example assumes that a mail merge data source is attached to the active publication.

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next intItem
    End With
End Sub
```




Show All

Comparison Property

Returns or sets an **MsoFilterComparison** constant that represents how to compare the **Column** and **CompareTo** properties. Read/write.

MsoFilterComparison can be one of these MsoFilterComparison constants.
msoFilterComparisonContains
msoFilterComparisonEqual
msoFilterComparisonGreaterThan
msoFilterComparisonGreaterThanEqual
msoFilterComparisonIsBlank
msoFilterComparisonIsNotBlank
msoFilterComparisonLessThan
msoFilterComparisonLessThanEqual
msoFilterComparisonNotContains
msoFilterComparisonNotEqual

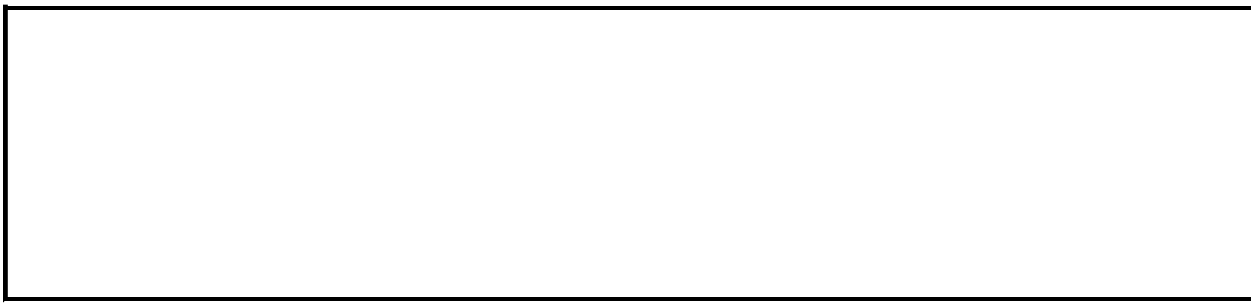
expression.**Comparison**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example changes an existing filter to remove from the mail merge all records that do not have a Region field equal to "WA". This example assumes that a mail merge data source is attached to the active publication.

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next intItem
    End With
End Sub
```




Show All

Conjunction Property

Returns or sets an **MsoFilterConjunction** constant that represents how a filter criterion relates to other filter criteria in the **MailMergeFilters** object. Read/write.

MsoFilterConjunction can be one of these MsoFilterConjunction constants.

msoFilterConjunctionAnd

msoFilterConjunctionOr

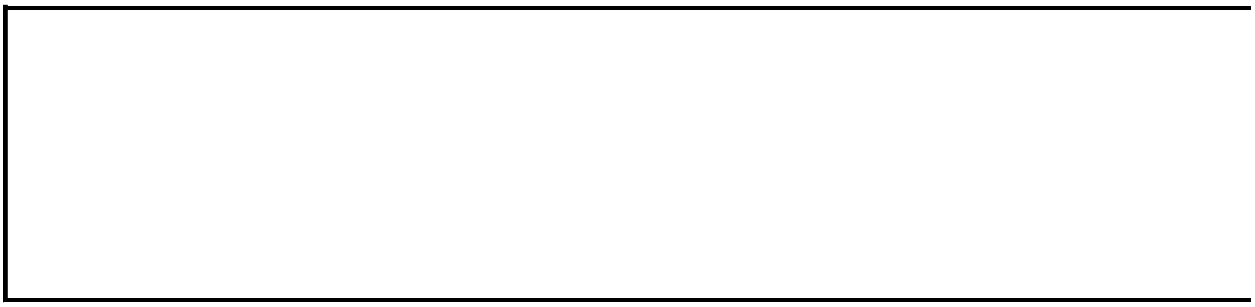
expression.**Conjunction**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example changes an existing filter to remove from the mail merge all records that do not have a Region field equal to "WA", and then adds the filter to the following filter, so that the the filter criteria must match both filters combined and not just one or the other.

```
Sub SetQueryCriterion()
    Dim intItem As Integer
    With ActiveDocument.MailMerge.DataSource.Filters
        For intItem = 1 To .Count
            With .Item(intItem)
                If .Column = "Region" Then
                    .Comparison = msoFilterComparisonNotEqual
                    .CompareTo = "WA"
                    If .Conjunction = "Or" Then .Conjunction = "And"
                End If
            End With
        Next
    End With
End Sub
```



ConnectionSiteCount Property

Returns a **Long** indicating the count of connection sites on the current **Shape** object. Read-only.

expression.**ConnectionSiteCount**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The number of connection sites varies depending on the shape geometry. Rectangular objects including tables and Web controls will most likely have four connection sites, one centered on each edge of the shape.

Example

This example adds two rectangles to the active publication and joins them with two connectors. The beginnings of both connectors attach to connection site one on the first rectangle; the ends of the connectors attach to the first and last connection sites of the second rectangle. Then it counts the number of connections on the first rectangle.

```
Sub Connections()

    Dim shpNew As Shapes
    Dim shpFirstRect As Shape
    Dim shpSecondRect As Shape
    Dim intLastSite As Integer
    Dim intCount As Integer

    Set shpNew = Application.ActiveDocument _
        .MasterPages(Item:=1).Shapes
    Set shpFirstRect = shpNew.AddShape(Type:=msoShapeRectangle, _
       Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpSecondRect = shpNew.AddShape(msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)
    varLastSite = shpSecondRect.ConnectionSiteCount

    ' Add the first connector from rectangle 1,
    ' site 1 to rectangle 2, site 1.
    With shpNew.AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
        .ConnectorFormat
        .BeginConnect ConnectedShape:=shpFirstRect, ConnectionSite:=
        .EndConnect ConnectedShape:=shpSecondRect, ConnectionSite:=1
    End With

    ' Add the second connector from rectangle 1,
    ' site 1 to rectangle 2, site 2.
    With shpNew.AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
        .ConnectorFormat
        .BeginConnect ConnectedShape:=shpFirstRect, ConnectionSite:=
        .EndConnect ConnectedShape:=shpSecondRect, _
           ConnectionSite:=intLastSite
    End With

    intCount = shpFirstRect.ConnectionSiteCount

End Sub
```


Show All

Connector Property

Returns an **MsoTriState** value indicating whether the specified shape is a connector. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shape is not a connector.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The shape is a connector.

expression.**Connector**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example deletes all connectors on page one of the active publication.

```
Dim i As Integer

With ActiveDocument.Pages(1).Shapes
    For i = .Count To 1 Step -1
        With .Item(i)
            If .Connector Then .Delete
        End With
    Next
End With
```

ConnectorFormat Property

Returns a **ConnectorFormat** object that contains connector formatting properties. Applies to **Shape** or **ShapeRange** objects that represent connectors.

expression.**ConnectorFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds two rectangles to the first page in the active publication and connects them with a curved connector.

```
Dim shpRect1 As Shape
Dim shpRect2 As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add two new rectangles.
    Set shpRect1 = .AddShape(Type:=msoShapeRectangle, _
        Left:=100, Top:=50, Width:=200, Height:=100)
    Set shpRect2 = .AddShape(Type:=msoShapeRectangle, _
        Left:=300, Top:=300, Width:=200, Height:=100)

    ' Add a new curved connector.
    With .AddConnector(Type:=msoConnectorCurve, _
            BeginX:=0, BeginY:=0, EndX:=100, EndY:=100) _
            .ConnectorFormat

        ' Connect the new connector to the two rectangles.
        .BeginConnect ConnectedShape:=shpRect1, ConnectionSite:=1
        .EndConnect ConnectedShape:=shpRect2, ConnectionSite:=1

        ' Reroute the connector to create the shortest path.
        .Parent.RerouteConnections
    End With

End With
```

ConnectString Property

Returns a **String** that represents the connection to the specified mail merge data source. Read-only.

expression.**ConnectString**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example checks if the connection string contains the characters OLEDB and displays a message accordingly.

```
Sub VerifyCorrectDataSource()

    With ActiveDocument.MailMerge.DataSource
        If InStr(.ConnectString, "OLEDB") > 0 Then
            MsgBox "OLE DB is used to connect to the data source."
        Else
            MsgBox "OLE DB is not used to connect to the data source
        End If
    End With

End Sub
```

ContainingObject Property

Returns an **Object** that represents the object that contains the text range. Read-only.

expression.**ContainingObject**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example returns the name of the object containing the specified text range.

```
Sub NameOfContainingObject()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.ContainingObject
        MsgBox The name of the object containing the text is " & .Na
    End With
End Sub
```

ContinueNumbersFromPreviousSectio Property

True if the specified section continues the numbering from the prvious section. Read/write **Boolean**.

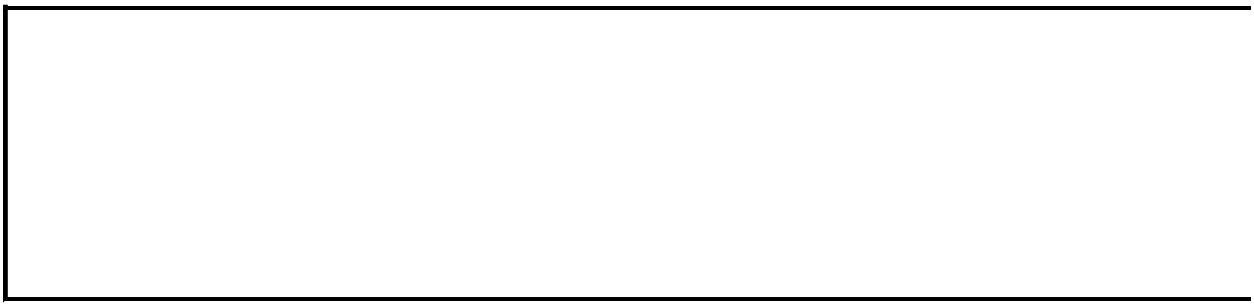
expression.**ContinueNumbersFromPreviousSection**

expression Required. An expression that returns a **Section** object.

Example

The following example adds three pages to the publication, adds a new section after the first page, and then sets the **ContinueNumbersFromPreviousSection** to **False** for the new section.

```
Dim objSection As Section
ActiveDocument.Pages.Add Count:=3, After:=1
Set objSection = ActiveDocument.Sections.Add(StartPageIndex:=2)
objSection.ContinueNumbersFromPreviousSection = False
```



Contrast Property

Returns or sets a **Single** indicating the contrast for the specified picture or OLE object. The value for this property must be a number from 0.0 (the least contrast) to 1.0 (the greatest contrast). Read/write.

expression.**Contrast**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **IncrementContrast** method to incrementally adjust the contrast from its current level.

Example

This example sets the contrast for the first shape in the active publication. The shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(1).PictureFormat _
    .Contrast = 0.8
```

Count Property

Returns a **Long** that represents the number of items in the specified collection. Read-only.

expression.**Count**

expression Required. An expression that returns one of the above objects.

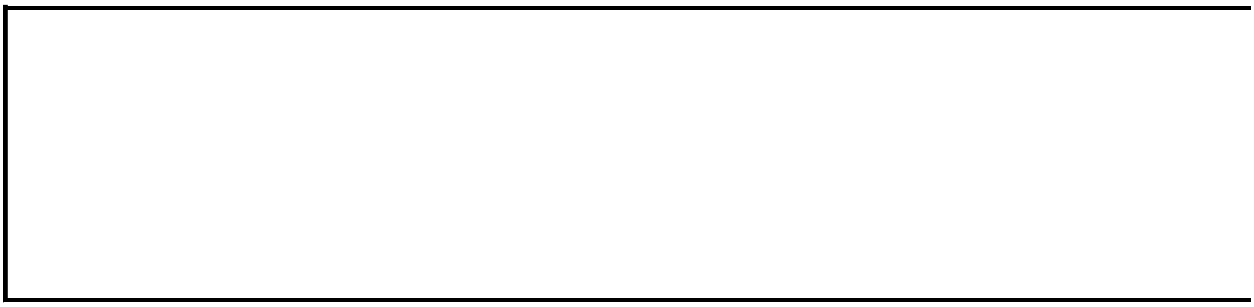
Example

This example displays the number of pages in the active document.

```
Sub CountNumberOfPages()
    MsgBox "Your publication contains " & _
        ActiveDocument.Pages.Count & " page(s)."
End Sub
```

This example displays the number of shapes in the active document.

```
Sub CountNumberOfShapes()
    Dim intShapes As Integer
    Dim pg As Page

    For Each pg In ActiveDocument.Pages
        intShapes = intShapes + pg.Shapes.Count
    Next

    MsgBox "Your publication contains " & intShapes & " shape(s)."
End Sub
```



Creator Property

Returns a **Long** that represents the application in which the specified object was created. For example, if the object was created in Microsoft Publisher, this property returns the hexadecimal number 4D505542, which represents the string "MSPB." This value can also be represented by the constant **wdCreatorCode**. Read-only.

expression.**Creator**

expression Required. An expression that returns one of the objects in the Applies To list.

CropBottom Property

Returns or sets a **Variant** indicating the amount by which the bottom edge of a picture or OLE object is cropped. Read/write.

expression.**CropBottom**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Negative values crop the bottom edge away from the center of the frame and positive values crop toward the top edge of the frame.

The valid range of crop values depends on the frame’s position and size. For an unrotated frame, the lowest negative value allowed is the distance between the bottom edge of frame and the bottom edge of the scratch area. The highest positive value allowed is the current frame height.

Cropping is calculated relative to the original size of the picture. For example, if you insert a picture that is originally 100 points high, rescale it so that it's 200 points high, and then set the **CropBottom** property to 50, 100 points (not 50) will be cropped off the bottom of your picture.

Use the **CropLeft** , **CropRight** , and **CropTop** properties to crop other edges of a picture or OLE object.

Example

This example crops 20 points off the bottom of the third shape in the active publication. For the example to work, the shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(3).PictureFormat _
    .CropBottom = 20
```

This example crops the percentage specified by the user off the bottom of the selected shape, regardless of whether the shape has been scaled. For the example to work, the selected shape must be either a picture or an OLE object.

```
Dim sngPercent As Single
Dim shpCrop As Shape
Dim sngPoints As Single
Dim sngHeight As Single

sngPercent = InputBox("What percentage do you " & _
    "want to crop off the bottom of this picture?")

Set shpCrop = Selection.ShapeRange(1)
With shpCrop.Duplicate
    .ScaleHeight Factor:=1, _
        RelativeToOriginalSize:=True
    sngHeight = .Height
    .Delete
End With

sngPoints = sngHeight * sngPercent / 100

shpCrop.PictureFormat.CropBottom = sngPoints
```

CropLeft Property

Returns or sets a **Variant** indicating the amount by which the left edge of a picture or OLE object is cropped. Read/write.

expression.**CropLeft**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Negative values crop the bottom edge away from the center of the frame and positive values crop toward the right edge of the frame.

The valid range of crop values depends on the frame’s position and size. For an unrotated frame, the lowest negative value allowed is the distance between the left edge of frame and the left edge of the scratch area. The highest positive value allowed is the current frame width.

Cropping is calculated relative to the original size of the picture. For example, if you insert a picture that is originally 100 points wide, rescale it so that it's 200 points wide, and then set the **CropLeft** property to 50, 100 points (not 50) will be cropped off the left of your picture.

Use the **CropRight** , **CropTop** , and **CropBottom** properties to crop other edges of a picture or OLE object.

Example

This example crops 20 points off the left of the third shape in the active publication. For the example to work, the shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(3).PictureFormat _
    .CropLeft = 20
```

This example crops the percentage specified by the user off the left of the selected shape, regardless of whether the shape has been scaled. For the example to work, the selected shape must be either a picture or an OLE object.

```
Dim sngPercent As Single
Dim shpCrop As Shape
Dim sngPoints As Single
Dim sngWidth As Single

sngPercent = InputBox("What percentage do you " & _
    "want to crop off the left of this picture?")

Set shpCrop = Selection.ShapeRange(1)
With shpCrop.Duplicate
    .ScaleWidth Factor:=1, _
        RelativeToOriginalSize:=True
    sngWidth = .Width
    .Delete
End With

sngPoints = sngWidth * sngPercent / 100

shpCrop.PictureFormat.CropLeft = sngPoints
```

CropRight Property

Returns or sets a **Variant** indicating the amount by which the right edge of a picture or OLE object is cropped. Read/write.

expression.**CropRight**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Negative values crop the bottom edge away from the center of the frame and positive values crop toward the left edge of the frame.

The valid range of crop values depends on the frame’s position and size. For an unrotated frame, the lowest negative value allowed is the distance between the right edge of frame and the right edge of the scratch area. The highest positive value allowed is the current frame width.

Cropping is calculated relative to the original size of the picture. For example, if you insert a picture that is originally 100 points wide, rescale it so that it's 200 points wide, and then set the **CropRight** property to 50, 100 points (not 50) will be cropped off the right of your picture.

Use the **CropLeft** , **CropTop** , and **CropBottom** properties to crop other edges of a picture or OLE object.

Example

This example crops 20 points off the right of the third shape in the active publication. For the example to work, the shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(3).PictureFormat _
    .CropRight = 20
```

This example crops the percentage specified by the user off the right of the selected shape, regardless of whether the shape has been scaled. For the example to work, the selected shape must be either a picture or an OLE object.

```
Dim sngPercent As Single
Dim shpCrop As Shape
Dim sngPoints As Single
Dim sngWidth As Single

sngPercent = InputBox("What percentage do you " & _
    "want to crop off the right of this picture?")

Set shpCrop = Selection.ShapeRange(1)
With shpCrop.Duplicate
    .ScaleWidth Factor:=1, _
        RelativeToOriginalSize:=True
    sngWidth = .Width
    .Delete
End With

sngPoints = sngWidth * sngPercent / 100

shpCrop.PictureFormat.CropRight = sngPoints
```

CropTop Property

Returns or sets a **Variant** indicating the amount by which the top edge of a picture or OLE object is cropped. Read/write.

expression.**CropTop**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Negative values crop the top edge away from the center of the frame and positive values crop toward the bottom edge of the frame.

The valid range of crop values depends on the frame’s position and size. For an unrotated frame, the lowest negative value allowed is the distance between the top edge of frame and the top edge of the scratch area. The highest positive value allowed is the current frame height.

Cropping is calculated relative to the original size of the picture. For example, if you insert a picture that is originally 100 points high, rescale it so that it's 200 points high, and then set the **CropTop** property to 50, 100 points (not 50) will be cropped off the top of your picture.

Use the **CropLeft** , **CropRight** , and **CropBottom** properties to crop other edges of a picture or OLE object.

Example

This example crops 20 points off the top of the third shape in the active publication. For the example to work, the shape must be either a picture or an OLE object.

```
ActiveDocument.Pages(1).Shapes(3).PictureFormat _
    .CropTop = 20
```

This example crops the percentage specified by the user off the top of the selected shape, regardless of whether the shape has been scaled. For the example to work, the selected shape must be either a picture or an OLE object.

```
Dim sngPercent As Single
Dim shpCrop As Shape
Dim sngPoints As Single
Dim sngHeight As Single

sngPercent = InputBox("What percentage do you " & _
    "want to crop off the top of this picture?")

Set shpCrop = Selection.ShapeRange(1)
With shpCrop.Duplicate
    .ScaleHeight Factor:=1, _
        RelativeToOriginalSize:=True
    sngHeight = .Height
    .Delete
End With

sngPoints = sngHeight * sngPercent / 100

shpCrop.PictureFormat.CropTop = sngPoints
```

CurrentValueId Property

Returns or sets a **Long** indicating the value of a setting in the specified publication design or Design Gallery object's wizard. Read/write.

expression.**CurrentValueId**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Accessing this property for a publication design setting whose **Enabled** property is **False** causes an error.

Example

The following example changes the settings of the current publication design (Newsletter Wizard) so that the publication has a region dedicated to the customer's address.

```
Dim wizTemp As Wizard
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp.Properties
    .FindPropertyById(ID:=901).CurrentValueId = 1
End With
```


Show All

Cyan Property

Sets or returns a **Long** that represents the cyan component of a CMYK color. Value can be any number between 0 and 255. Read/write.

expression.**Cyan**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two new shapes and then sets the CMYK fill color for one shape and sets the CMYK values of the second shape to the same CMYK values.

```
Sub ReturnAndSetCMYK()
    Dim lngCyan As Long
    Dim lngMagenta As Long
    Dim lngYellow As Long
    Dim lngBlack As Long
    Dim shpHeart As Shape
    Dim shpStar As Shape

    Set shpHeart = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeHeart, Left:=100, _
        Top:=100, Width:=100, Height:=100)
    Set shpStar = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=100, Width:=150, Height:=150)

    With shpHeart.Fill.ForeColor.CMYK
       .SetCMYK 10, 80, 200, 30
        lngCyan = .Cyan
        lngMagenta = .Magenta
        lngYellow = .Yellow
        lngBlack = .Black
    End With

    'Sets new shape to current shape's CMYK colors
    shpStar.Fill.ForeColor.CMYK.SetCMYK _
        Cyan:=lngCyan, Magenta:=lngMagenta, _
        Yellow:=lngYellow, Black:=lngBlack
End Sub
```


Show All

DashStyle Property

Returns or sets an **MsoLineDashStyle** constant indicating the dash style for the specified line. Read/write.

MsoLineDashStyle can be one of these MsoLineDashStyle constants.

msoLineDash

msoLineDashDot

msoLineDashDotDot

msoLineDashStyleMixed Return value only; indicates a combination of the other states in the specified shape range.

msoLineLongDash

msoLineLongDashDot

msoLineRoundDot

msoLineSolid

msoLineSquareDot

expression.**DashStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a blue dashed line to the active publication.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=10, BeginY:=10, _
        EndX:=250, EndY:=250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```


Show All

DataFieldName Property

Returns or sets a **String** which represents the name of the field in the mail merge data source to which a mapped data field maps. An empty string is returned if the specified data field is not mapped to a mapped data field. Read/write.

expression.**DataFieldName**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a table on a new page of the current publication and lists the mapped data fields available and the fields in the data source to which they are mapped. This example assumes that the current publication is a mail merge publication and that the data source fields have corresponding mapped data fields.

```
Sub MappedFields()
    Dim intCount As Integer
    Dim intRows As Integer
    Dim docPub As Document
    Dim pagNew As Page
    Dim shpTable As Shape
    Dim tblTable As Table
    Dim rowTable As Row

    On Error Resume Next

    Set docPub = ThisDocument
    Set pagNew = ThisDocument.Pages.Add(Count:=1, After:=1)
    intRows = docPub.MailMerge.DataSource.MappedDataFields.Count + 1

    'Creates new table with a heading row
    Set shpTable = pagNew.Shapes.AddTable(NumRows:=intRows, _
        numColumns:=2, Left:=100, Top:=100, Width:=400, Height:=12)
    Set tblTable = shpTable.Table
    With tblTable.Rows(1)
        With .Cells(1).Text
            .Text = "Mapped Data Field"
            .Font.Bold = msoTrue
        End With
        With .Cells(2).Text
            .Text = "Data Source Field"
            .Font.Bold = msoTrue
        End With
    End With

    With docPub.MailMerge.DataSource
        For intCount = 2 To intRows - 1
            'Inserts mapped data field name and the
            'corresponding data source field name
            tblTable.Rows(intCount - 1).Cells(1).Text _
                .Text = .MappedDataFields(Index:=intCount).Name
            tblTable.Rows(intCount - 1).Cells(2).Text _
                .Text = .MappedDataFields(Index:=intCount).DataFiel
```

```
        Next
    End With
End Sub
```

DataFields Property

Returns a **MailMergeDataFields** collection that represents the fields in the specified data source.

expression.**DataFields**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the value of the value of the FirstName and LastName fields from the active record in the data source attached to the active publication.

```
Sub ShowNameForActiveRecord()
    Dim mdfFirst As MailMergeDataField
    Dim mdfLast As MailMergeDataField

    With ActiveDocument.MailMerge.DataSource
        Set mdfFirst = .DataFields.Item("FirstName")
        Set mdfLast = .DataFields.Item("LastName")
        MsgBox "The active record in the attached " & _
            vbLf & "data source is : " & _
            mdfFirst.Value & " " & _
            mdfLast.Value
    End With
End Sub
```


Show All

DataFileFormat Property

Sets or returns a **PbSubmitDataFormatType** constant that represents the format to use when saving Web form data to a file. Read/write.

PbSubmitDataFormatType can be one of these PbSubmitDataFormatType constants.

pbSubmitDataFormatCSV Saves Web form data to a comma-delimited text file.

pbSubmitDataFormatHTML Saves Web form data to an HTML file.

pbSubmitDataFormatRichText Saves Web form data to a formatted file.

pbSubmitDataFormatTab Saves Web form data to a tab-delimited text file.

expression.**DataFileFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets Publisher to process Web form data by saving it to a comma-delimited text file on the same Web server as the form is stored. (Note that *Filename* must be replaced with a valid file name for this example to work.)

```
Sub WebDataFile()
    With ThisDocument.Pages(1).Shapes(1).WebCommandButton
        .DataRetrievalMethod = pbSubmitDataRetrievalSaveOnServer
        .DataFileFormat = pbSubmitDataFormatCSV
        .DataFileName = "Filename"
    End With
End Sub
```

DataFileName Property

Returns or sets a **String** that represents the name of the file in which to save data from a Web form. Read/write.

expression.**DataFileName**

expression Required. An expression that returns a **WebCommandButton** object.

Example

This example sets Publisher to process Web form data by saving it to a comma-delimited text file on the same Web server as the form is stored.

```
Sub WebDataFile()
    With ThisDocument.Pages(1).Shapes(1).WebCommandButton
        .DataRetrievalMethod = pbSubmitDataRetrievalSaveOnServer
        .DataFileFormat = pbSubmitDataFormatCSV
        .DataFileName = "WebFormData.txt"
    End With
End Sub
```


Show All

DataRetrievalMethod Property

Sets or returns a **PbSubmitDataRetrievalMethodType** that represents the way data from a Web form is processed. Read/write.

PbSubmitDataRetrievalMethodType can be one of these PbSubmitDataRetrievalMethodType constants.

pbSubmitDataRetrievalEmail Processes form data by sending an e-mail message to a specified e-mail address.

pbSubmitDataRetrievalProgram Processes form data using a script program provided by your Internet Service Provider.

pbSubmitDataRetrievalSaveOnServer Saves form data to a file stored on your Web server.

expression.DataRetrievalMethod

expression Required. An expression that returns a **WebCommandButton** object.

Example

This example sets Publisher to process data on the Web form in the current publication by sending an e-mail message to a specified e-mail address.

```
Sub WebFormData()
    With ThisDocument.Pages(1).Shapes(1).WebCommandButton
        .DataRetrievalMethod = pbSubmitDataRetrievalEmail
        .EmailAddress = "someone@example.com"
        .EmailSubject = "Web form data"
    End With
End Sub
```


Show All

DataSource Property

Returns a **MailMergeDataSource** object that refers to the data source attached to a mail merge or catalog merge main publication.

expression.**DataSource**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the path and file name of the data source attached to the active publication.

```
Sub DataSourceName()
    With ActiveDocument.MailMerge.DataSource
        If .Name <> "" Then _
            MsgBox "The path and filename of the " & _
            "attached data source is : " & vbCr & .Name
    End With
End Sub
```


Show All

DefaultPubDirection Property

Returns or sets a **PbDirectionType** constant that represents the default direction in which text flows when a new publication is created. Read/write.

PbDirectionType can be one of these PbDirectionType constants.
pbDirectionLeftToRight
pbDirectionRightToLeft

expression.**DefaultPubDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property generates an error if you are not running a bi-directional-enabled version of Microsoft Publisher (for example, Arabic).

Example

This example sets the default direction for new publications and text flow in a bi-directional-enabled version of Publisher.

```
Sub SetDefaultDirection()
    With Options
        .DefaultPubDirection = pbDirectionRightToLeft
        .DefaultTextFlowDirection = pbDirectionRightToLeft
    End With
End Sub
```

DefaultTabStop Property

Returns or sets a **Variant** corresponding to the default tab stop for all text in the active publication. Valid range is 1 to 1584 points (0.014" to 22"). Once set, numeric values are considered to be in points. **String** values may be in any unit supported by Microsoft Publisher. Point values are always returned. If values are outside the valid range, an error is returned. Read/write.

expression.**DefaultTabStop**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **InchesToPoints** method to convert inches to points.

Example

This example sets the **DefaultTabStop** property to 72 points for all text in the active publication.

```
Sub SetTab()

    Application.ActiveDocument.DefaultTabStop = 72

End Sub
```

DefaultText Property

Returns or sets a **String** that represents the default text in a Web text box control. Read/write.

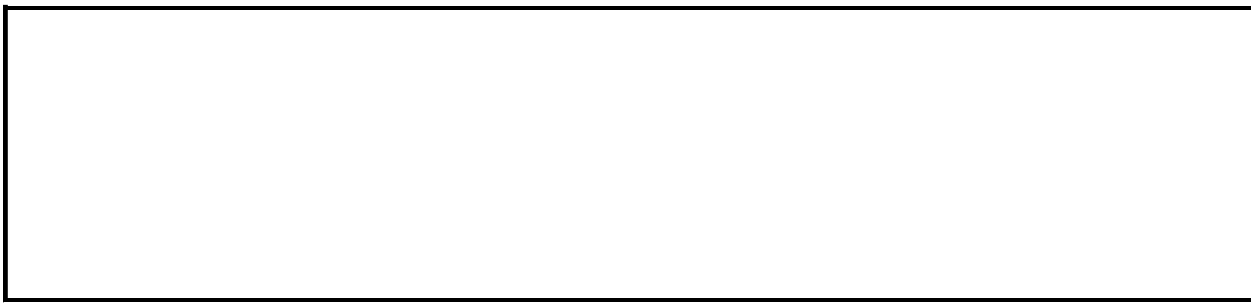
expression.**DefaultText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web text box control in the active publication, sets the default text and the character limit for the text box, and specifies that it is a required control.

```
Sub AddWebTextBoxControl()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlMultiLineTextBox, Left:=72, _
            Top:=72, Width:=300, Height:=100).WebTextBox
        .DefaultText = "Please enter text here."
        .Limit = 200
        .RequiredControl = msoTrue
    End With
End Sub
```




Show All

DefaultTextFlowDirection Property

Returns or sets a **PbDirectionType** constant that represents a global Microsoft Publisher option, indicating whether text flows from left to right or from right to left in a publication. Read/write.

PbDirectionType can be one of these PbDirectionType constants.

pbDirectionLeftToRight

pbDirectionRightToLeft

expression.**DefaultTextFlowDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property generates an error if you are not running a bi-directional-enabled version of Publisher (for example, Arabic).

Example

This example sets the default direction for new publications and text flow in a bi-directional-enabled version of Publisher.

```
Sub SetDefaultDirection()
    With Options
        .DefaultPubDirection = pbDirectionRightToLeft
        .DefaultTextFlowDirection = pbDirectionRightToLeft
    End With
End Sub
```

Depth Property

Returns or sets a **Variant** indicating the depth of the shape's extrusion. Read/write.

expression.**Depth**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Positive values produce an extrusion whose front face is the original shape; negative values produce an extrusion whose back face is the original shape. The valid range is – 600 through 9600 points, or the equivalent distance in all other units.

Example

This example adds an oval to the active publication, and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Dim shpNew As Shape

Set shpNew = ActiveDocument.Pages(1).Shapes _
    .AddShape(Type:=msoShapeOval, _
    Left:=90, Top:=90, Width:=90, Height:=40)

With shpNew.ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)
End With
```


Show All

Description Property

Description property as it applies to the **TextStyle** object.

Returns a **String** that represents the description of the specified style. For example, a typical description for the Normal style might be "(Default) Times New Roman, (Asian) MS Mincho, 10 pt, Main (Black), Kerning 14 pt, Left, Line spacing 1 sp." Read-only.

expression.**Description**

expression Required. An expression that returns one of the objects in the Applies To list.

Description property as it applies to the **WebPageOptions** object.

Returns or sets a **String** that represents the description of a Web page within a Web publication. Read/write.

expression.**Description**

expression Required. An expression that returns a **WebPageOptions** object.

Example

As it applies to the **TextStyle** object.

This example displays the description for the Normal style.

```
Sub ShowStyleDescription()
    MsgBox "The Normal style has the following formatting attributes
        vbLf & ActiveDocument.TextStyles("Normal").Description
End Sub
```

As it applies to the **WebPageOptions** object.

This example sets the description for page two of the active Web publication.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(2).WebPageOptions

With theWPO
    .Description = "Company Profile"
End With
```


Show All

Design Property

Sets or returns a **PbWizardNavBarDesign** constant representing the design of the specified Web navigation bar set. Read/write.

The **Design** property can be any of these **PbWizardNavBarDesign** constants:

pbnbDesignAmbient
pbnbDesignBaseline
pbnbDesignBracket
pbnbDesignBulletStaff
pbnbDesignCapsule
pbnbDesignCornice
pbnbDesignCounter
pbnbDesignDimension
pbnbDesignDottedArrow
pbnbDesignEdge
pbnbDesignEnclosedArrow
pbnbDesignEndCap
pbnbDesignHollowArrow
pbnbDesignKeyPunch
pbnbDesignOffset
pbnbDesignOutline
pbnbDesignRadius
pbnbDesignRectangle
pbnbDesignRoundBullet
pbnbDesignSquareBullet
pbnbDesignStaff
pbnbDesignTopBar
pbnbDesignTopDrawer
pbnbDesignTopLine
pbnbDesignUnderscore
pbnbDesignWatermark

expression.**Design**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Example

This example adds a new Web navigation bar set to every page in the active document, sets the button style to large, and then sets the design property to **pbnbDesignCapsule**.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets.AddSet(Name:="ne
With objWebNav
    .AddToEveryPage Left:=10, Top:=10
    .ButtonStyle = pbnbButtonStyleLarge
    .Design = pbnbDesignCapsule
End With
```

DiacriticColor Property

Returns a **ColorFormat** object representing the 24-bit color used for diacritics in a right-to-left language publication.

expression.**DiacriticColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the text in the first story of the current publication to see if its color is red and it is formatted right-to-left.

```
Sub FontDiColor()

    Dim fntDiColor As Font

    Set fntDiColor = Application.ActiveDocument. _
        Stories(1).TextRange.Font

    If fntDiColor.UseDiacriticColor = msoTrue And _
            fntDiColor.DiacriticColor.RGB = RGB(255, 0, 0) Then
        MsgBox "Your text is red"
    Else
        MsgBox "This is not a right-to-left language" _
        & " or your color is not red"
    End If

End Sub
```


Show All

Diagonal Property

Sets or returns a **PbCellDiagonalType** constant that represents a cell that is diagonally split. Read/write.

PbCellDiagonalType can be one of these PbCellDiagonalType constants.
pbTableCellDiagonalDown
pbTableCellDiagonalMixed
pbTableCellDiagonalNone
pbTableCellDiagonalUp

expression.**Diagonal**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a page to the active publication, creates a table on that new page, and diagonally splits all cells in even-numbered columns.

```
Sub CreateNewTable()

    Dim pgeNew As Page
    Dim shpTable As Shape
    Dim tblNew As Table
    Dim celTable As Cell
    Dim rowTable As Row

    'Creates a new document with a five-row by five-column table
    Set pgeNew = ActiveDocument.Pages.Add(Count:=1, After:=1)
    Set shpTable = pgeNew.Shapes.AddTable(NumRows:=5, NumColumns:=5,
        Left:=72, Top:=72, Width:=468, Height:=100)
    Set tblNew = shpTable.Table

    'Inserts a diagonal split into all cells in even-numbered column
    For Each rowTable In tblNew.Rows
        For Each celTable In rowTable.Cells
            If celTable.Column Mod 2 = 0 Then
                celTable.Diagonal = pbTableCellDiagonalUp
            End If
        Next celTable
    Next rowTable

End Sub
```

DisplayPrintTroubleshooter Property

True to automatically display a Help topic to troubleshoot printing problems when printing publications. Read/write **Boolean**.

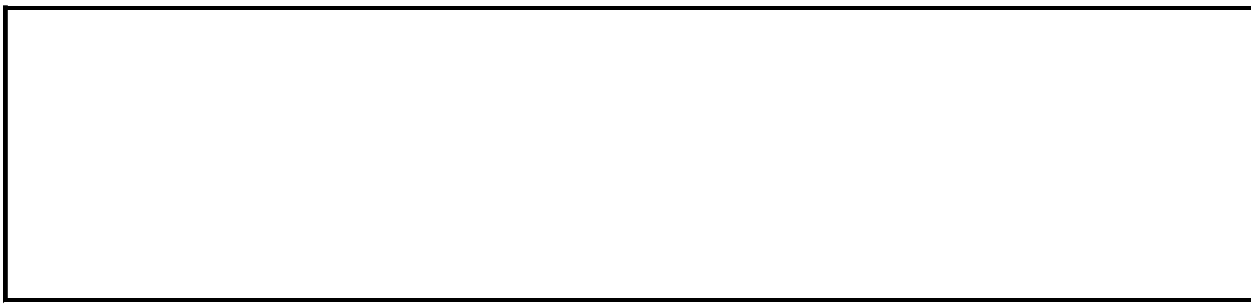
expression.**DisplayPrintTroubleshooter**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enables displaying the Print Troubleshooting Help topic when printing publications.

```
Sub ShowPrinterHelp()
    Application.Options.DisplayPrintTroubleshooter = True
End Sub
```



DisplayStatusBar Property

True for Microsoft Publisher to show the status bar at the bottom of the Publisher window. Read/write **Boolean**.

expression.**DisplayStatusBar**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example hides the status bar from view.

```
Sub HideStatusBar()
    Options.DisplayStatusBar = False
End Sub
```


Show All

DistanceAuto Property

Returns or sets an **MsoTriState** constant indicating whether an appropriate distance between an inline shape and any surrounding text is automatically calculated. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shape's edges are not adjusted depending on the margins of the text box it overlaps.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Set value that toggles the property value between **msoTrue** and **msoFalse**.

msoTrue *default* The shape's edges are automatically adjusted depending on the margins of the text box it overlaps.

expression.**DistanceAuto**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets shape one on page one of the active publication so that its edges are not automatically adjusted based on its distance from surrounding text.

```
Sub SetDistanceAutoProperty()
    With ActiveDocument.Pages(1).Shapes(1).TextWrap
        .Type = pbWrapTypeSquare
        .DistanceAuto = msoFalse
    End With
End Sub
```

DistanceBottom Property

When the **Type** property of the **WrapFormat** object is set to **pbWrapTypeSquare**, returns or sets a **Variant** that represents the distance (in points) between the document text and the bottom edge of the specified shape. Read/write.

expression.**DistanceBottom**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active document and specifies that the document text wrap around the left and right sides of the square that circumscribes the oval. The example sets a 0.1-inch margin between the document text and the top, bottom, left side, and right side of the square.

```
Sub AddNewShape()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, Left:=36, _
        Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```

DistanceLeft Property

When the **Type** property of the **WrapFormat** object is set to **pbWrapTypeSquare**, returns or sets a **Variant** that represents the distance (in points) between the document text and the left edge of the specified shape. Read/write.

expression.**DistanceLeft**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active document and specifies that the document text wrap around the left and right sides of the square that circumscribes the oval. The example sets a 0.1-inch margin between the document text and the top, bottom, left side, and right side of the square.

```
Sub AddNewShape()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, Left:=36, _
        Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```

DistanceRight Property

When the **Type** property of the **WrapFormat** object is set to **pbWrapTypeSquare**, returns or sets a **Variant** that represents the distance (in points) between the document text and the right edge of the specified shape. Read/write.

expression.**DistanceRight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active document and specifies that the document text wrap around the left and right sides of the square that circumscribes the oval. The example sets a 0.1-inch margin between the document text and the top, bottom, left side, and right side of the square.

```
Sub AddNewShape()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, Left:=36, _
        Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```

DistanceTop Property

When the **Type** property of the **WrapFormat** object is set to **pbWrapTypeSquare**, returns or sets a **Variant** that represents the distance (in points) between the document text and the top edge of the specified shape. Read/write.

expression.**DistanceTop**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active document and specifies that the document text wrap around the left and right sides of the square that circumscribes the oval. The example sets a 0.1-inch margin between the document text and the top, bottom, left side, and right side of the square.

```
Sub AddNewShape()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, Left:=36, _
        Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```


Show All

DocumentDirection Property

Returns or sets a **PbDirectionType** constant that indicates whether text in the document is read from left to right or from right to left. Read/write.

PbDirectionType can be one of these PbDirectionType constants.
pbDirectionLeftToRight
pbDirectionRightToLeft

expression.**DocumentDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **DocumentDirection** property affects the way the document is read but not the flow of text in the document. For example, if the document has a binding edge and is printed on both sides of the page, the binding edge for a left-to-right document would be different from the binding edge of a right-to-left document.

To format the direction of text flow, use the **DefaultTextFlowDirection** property to specify the default text flow for the entire document, or use the **Orientation** property for an individual text frame to specify a text flow direction other than the default for the specified text frame only.

Example

This example sets the active publication to read from left to right.

```
Sub SetBiDiText()
    ActiveDocument.DocumentDirection = pbDirectionRightToLeft
End Sub
```

Documents Property

Returns a **Documents** collection that represents all open publications. Read-only.

expression.**Documents**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example lists all of the open publications.

```
Dim objDocument As Document
Dim strMsg As String
For Each objDocument In Documents
    strMsg = strMsg & objDocument.Name & vbCrLf
Next objDocument
MsgBox Prompt:=strMsg, Title:="Current Documents Open", Buttons:=vbO
```

DocumentUpdating Property

Returns or sets a **Boolean** indicating whether the screen is updated while mail merge code is running. Default is **True** (the screen is updated). Read/write.

expression.**DocumentUpdating**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Turning document updating off during run time can speed execution of Visual Basic code. However, it is recommended to provide some indication of status so that the user is aware that the program is functioning correctly.

Example

The following example turns off document updating at the beginning of a mail merge subroutine and turns it back on at the end of the subroutine.

```
Sub MailMergeProcedure()
    ActiveDocument.MailMerge.DocumentUpdating
= False

    ' Mail merge code.


ActiveDocument.MailMerge.DocumentUpdating
= True
End Sub
```

DragAndDropText Property

True to enable dragging and dropping of text. Read/write **Boolean**.

expression.**DragAndDropText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets global options for Microsoft Publisher, including enabling dragging and dropping to reposition text.

```
Sub SetGlobalOptions()
    With Options
        .AutoFormatWord = True
        .AutoKeyboardSwitching = True
        .AutoSelectWord = True
        .DragAndDropText = True
        .UseCatalogAtStartup = False
        .UseHelpfulMousePointers = False
    End With
End Sub
```

Drop Property

For callouts with an explicitly set drop value, this property returns the vertical distance from the edge of the text bounding box to the place where the callout line attaches to the text box. This distance is measured from the top of the text box unless the **AutoAttach** property is set to **True** and the text box is to the left of the origin of the callout line (where the callout points). In this case, the drop distance is measured from the bottom of the text box. Read-only **Variant**.

expression.**Drop**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Use the **CustomDrop** method to set the value of this property.

The value of this property accurately reflects the position of the callout line attachment to the text box only if the callout has an explicitly set drop value — that is, if the value of the **DropType** property is **msoCalloutDropCustom**.

Example

This example replaces the custom drop for the first shape in the active publication with one of two preset drops, depending on whether the custom drop value is greater than or less than half the height of the callout text box. For the example to work, the shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    If .DropType = msoCalloutDropCustom Then
        If .Drop < .Parent.Height / 2 Then
            .PresetDrop DropType:=msoCalloutDropTop
        Else
            .PresetDrop DropType:=msoCalloutDropBottom
        End If
    End If
End With
```

DropCap Property

Returns a **DropCap** object that represents a dropped capital letter for the paragraphs in the specified text frame.

expression.**DropCap**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example applies a custom dropped capital that is three lines high and spans the first three characters of each paragraph in the specified text frame.

```
Sub SetDropCap()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange
        .DropCap.ApplyCustomDropCap FontName:="Snap ITC", _
            Bold:=True, Size:=3, Span:=3
        With .ParagraphFormat
            .SpaceBefore = 6
            .SpaceAfter = 6
        End With
    End With
End Sub
```


Show All

DropType Property

Returns an **MsoCalloutDropType** constant indicating where the callout line attaches to the callout text box. Read-only.

MsoCalloutDropType can be one of these MsoCalloutDropType constants.

msoCalloutDropBottom

msoCalloutDropCenter

msoCalloutDropCustom

msoCalloutDropMixed Indicates a combination of the other states in the specified shape range.

msoCalloutDropTop

expression.**DropType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the callout drop type is **msoCalloutDropCustom**, the values of the **Drop** and **AutoAttach** properties and the relative positions of the callout text box and callout line origin (where the callout points) are used to determine where the callout line attaches to the text box.

Use the **PresetDrop** method to set the value of this property.

Example

This example replaces the custom drop for the first shape in the active publication with one of two preset drops, depending on whether the custom drop value is greater than or less than half the height of the callout text box. For the example to work, the shape must be a callout.

```
With ActiveDocument.Pages(1).Shapes(1).Callout
    If .DropType = msoCalloutDropCustom Then
        If .Drop < .Parent.Height / 2 Then
            .PresetDrop DropType:=msoCalloutDropTop
        Else
            .PresetDrop DropType:=msoCalloutDropBottom
        End If
    End If
End With
```

Duplicate Property

Returns a **TextRange** object that represents a duplicate of the specified text range.

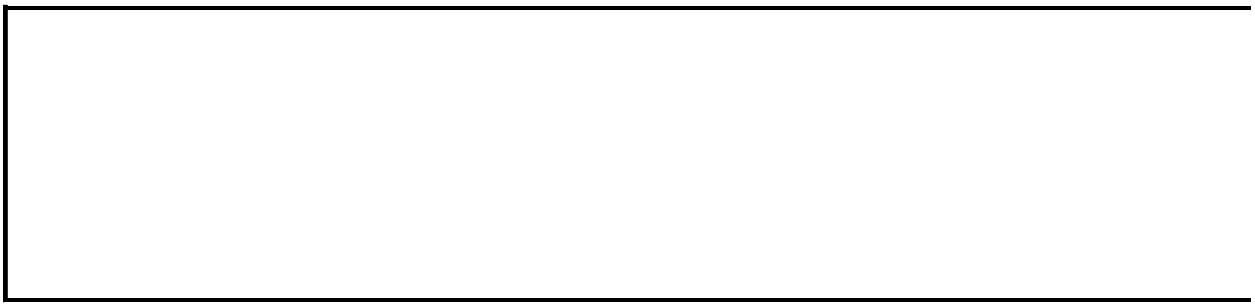
expression.**Duplicate**

expression Required. An expression that returns a **TextRange** object.

Example

This example sets the value of a string variable to the contents of the specified text box on the first page of the active publication. Then it creates a new page with a text box and sets the contents of the new text box equal to the value of the string variable.

```
Sub DuplicateTextBoxContents()
    Dim strDuplicate As String
    Dim pagNew As Page

    With ThisDocument.Pages(1).Shapes(1).TextFrame.TextRange
        strDuplicate = .Duplicate
    End With

    Set pagNew = ThisDocument.Pages.Add(Count:=1, After:=1)

    pagNew.Shapes.AddTextbox(Orientation:=pbTextOrientationHorizonta
            Left:=72, Top:=72, Width:=200, Height:=200).TextFrame _
            .TextRange.Text = strDuplicate
End Sub
```




Show All

EchoAsterisks Property

MsoTrue if asterisks should be displayed in place of text that is entered into a Web text box control. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse Displays the text entered into a Web text box control.

msoTriStateMixed

msoTriStateToggle

msoTrue Displays asterisks in place of text entered into a Web text box control.

expression.**EchoAsterisks**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a Web text box control, sets the maximum limit as ten characters, specifies that entry is required, and masks the entry with asterisks when a user enters into the control.

```
Sub AddPasswordTextBox()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlSingleLineTextBox, Left:=100, _
            Top:=100, Width:=72, Height:=15)
        .Name = "Password"
        With .WebTextBox
            .Limit = 10
            .EchoAsterisks = msoTrue
            .RequiredControl = msoTrue
        End With
    End With
End Sub
```


Show All

EditingType Property

If the specified node is a vertex, this property returns an **MsoEditingType** constant indicating how changes made to the node affect the two segments connected to the node. If the node is a control point for a curved segment, this property returns the editing type of the adjacent vertex. Read-only.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto An automatically adjusted node.

msoEditingCorner A corner node.

msoEditingSmooth A smooth curve node.

msoEditingSymmetric A symmetric curve node.

expression.**EditingType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **SetEditingType** method to set the value of this property.

Example

This example changes all corner nodes to smooth curve nodes in the third shape in the active publication. The shape must be a freeform drawing.

```
Dim intNode As Integer

With ActiveDocument.Pages(1).Shapes(3).Nodes
    For intNode = 1 to .Count
        If .Item(intNode).EditingType = msoEditingCorner Then
            .SetEditingType Index:=intNode, _
                EditingType:=msoEditingSmooth
        End If
    Next
End With
```

EffectiveResolution Property

Returns a **Long** that represents, in dots per inch (dpi), the effective resolution of the picture. Read-only.

expression.**EffectiveResolution**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

The effective resolution of a picture is inversely proportional to the scaling at which the picture is printed. The larger the scaling, the lower the effective resolution. For example, suppose a picture measuring 4 inches by 4 inches was originally scanned at 300 dpi. If that picture is scaled to 2 inches by 2 inches, its effective resolution is 600 dpi.

Use the **OriginalResolution** property of the **PictureFormat** object to determine the resolution of linked pictures or OLE objects. Use the **HorizontalScale** and **VerticalScale** properties to determine the scaling of a picture.

Example

The following example returns a list of pictures whose effective resolution falls below a specified threshold (100 dpi) in the active publication.

```
Sub ListLowResolutionPictures()
        Dim pgLoop As Page
        Dim shpLoop As Shape

    For Each pgLoop In ActiveDocument.Pages
        For Each shpLoop In pgLoop.Shapes

            If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPi

                With shpLoop.PictureFormat
                    If .IsEmpty = msoFalse Then
                        If .EffectiveResolution < 100 Then
                            Debug.Print .Filename
                            Debug.Print "Page " & pgLoop.PageNumber
                            Debug.Print "Resolution in publication:
                        End If
                    End If
                End With

            End If

        Next shpLoop
    Next pgLoop

End Sub
```

EmailAddress Property

Sets or returns a **String** representing the e-mail address to use when processing Web form data. Read/write.

expression.**EmailAddress**

expression Required. An expression that returns a **WebCommandButton** object.

Example

This example sets Publisher to process data on the Web form in the current publication by sending an e-mail message to a specified e-mail address.

```
Sub WebFormData()
    With ThisDocument.Pages(1).Shapes(1).WebCommandButton
        .DataRetrievalMethod = pbSubmitDataRetrievalEmail
        .EmailAddress = "someone@example.com"
        .EmailSubject = "Web form data"
    End With
End Sub
```

EmailAsImg Property

True to send the entire publication page as a single JPEG image. Read/write **Boolean**.

expression.**EmailAsImg**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property increase your message's compatibility with older e-mail clients, but may result in larger file size.

This property is accessible for print publications in addition to Web publications.

The properties of the **WebOptions** object are used to specify the behavior of Web publications. This means that when any of these properties are modified, newly created Web publications will inherit the modified properties.

This property corresponds to the check box in the **E-Mail Options** section of the **Web** tab of the **Options** dialog box.

Example

The following example sets Publisher to e-mail publication pages as JPEG images.

```
Application.WebOptions.EmailAsImg = True
```

EmailSubject Property

Sets or returns a **String** that represents the subject for e-mail messages generated to process Web form data. Read/write.

expression.**EmailSubject**

expression Required. An expression that returns a **WebCommandButton** object.

Example

This example sets Publisher to process data on the Web form in the current publication by sending an e-mail message with a subject line to a specified e-mail address.

```
Sub WebFormData()
    With ThisDocument.Pages(1).Shapes(1).WebCommandButton
        .DataRetrievalMethod = pbSubmitDataRetrievalEmail
        .EmailAddress = "someone@example.com"
        .EmailSubject = "Web form data"
    End With
End Sub
```


Show All

Emboss Property

Returns or sets an **MsoTriState** constant indicating whether the specified text is formatted as embossed. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse The specified text is not formatted as embossed.

msoTriStateMixed Return value which indicates a combination of **msoTrue** and **msoFalse** for the specified text range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue The specified text is formatted as embossed.

expression.**Emboss**

expression Required. An expression that returns one of the objects in the Applies To list.

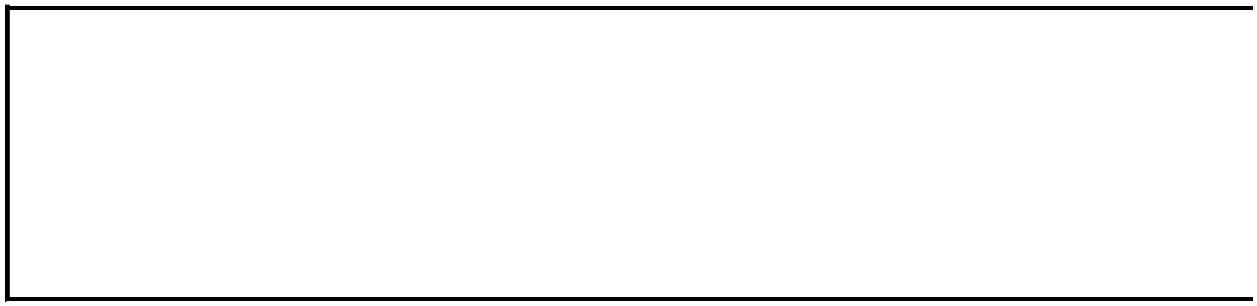
Remarks

Setting **Emboss** to **msoTrue** sets **Engrave** to **msoFalse** and vice versa.

Example

This example embosses all the text in the first story.

```
Sub FontEmb()

    Dim fntEmb As Font

    Set fntEmb = Application.ActiveDocument. _
        Stories(1).TextRange.Font
    If fntEmb.Emboss = msoFalse Or msoTriStateMixed Then
        fntEmb.Emboss = msoTrue
    End If

End Sub
```



Enabled Property

True if a wizard property is enabled. Read-only **Boolean**.

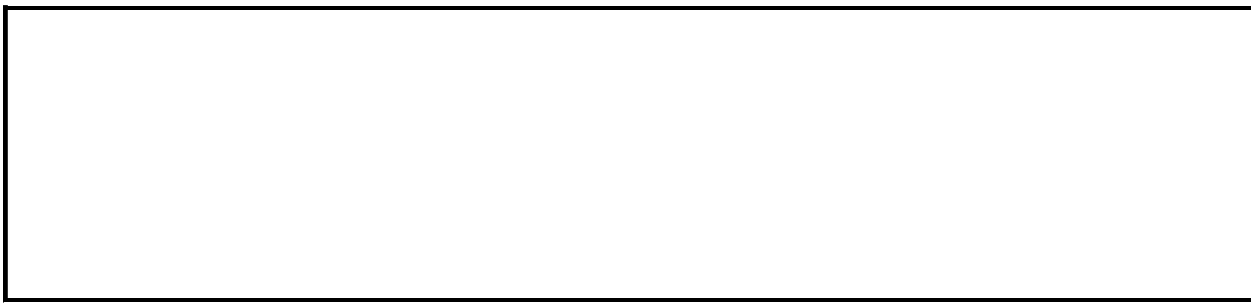
expression.**Enabled**

expression Required. An expression that returns a **WizardProperty** object.

Example

This example displays the name of each enabled wizard property in the active publication.

```
Sub SetEnabledProperty()
    Dim wizProperty As WizardProperty
    For Each wizProperty In ActiveDocument.Wizard.Properties
        If wizProperty.Enabled = True Then
            MsgBox "The name of the wizard property is " & wizProper
        End If
    Next
End Sub
```



EnableIncrementalUpload Property

Returns or sets a **Boolean** value that specifies whether changes made to a Web publication can be uploaded to a Web server independent of the entire publication. If **True**, only changes made to a publication will be uploaded to the Web server when published. If **False**, the entire publication will be uploaded to the Web server. The default value is **True**. Read/write.

expression.**EnableIncrementalUpload**

expression Required. An expression that returns a **WebOptions** object.

Remarks

The **EnableIncrementalUpload** property applies only to Web publications that have already been published to a Web server. If a Web publication has not already been published to a Web server, the entire publication will be published to the server during the initial publishing process, regardless of whether the **EnableIncrementalUpload** property is set to **True**. If a Web publication has already been published to a Web server and the **EnableIncrementalUpload** property is then set to **True**, only changes made to the Web publication, and not the entire publication, after this point will be published to the server.

Example

The following example tests whether the Web publication is set to upload only changes made to the publication. If not, the **EnableIncrementalUpload** property is set to **True** to specify that only changes to the publication be uploaded to the Web server.

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    If .EnableIncrementalUpload = False Then
        .EnableIncrementalUpload = True
    End If
End With
```


Show All

Encoding Property

Returns an **MsoEncoding** constant that specifies the encoding of the Web publication. Read/write.

MsoEncoding can be one of these MsoEncoding constants.

msoEncodingArabic
msoEncodingArabicASMO
msoEncodingArabicAutoDetect
msoEncodingArabicTransparentASMO
msoEncodingAutoDetect
msoEncodingBaltic
msoEncodingCentralEuropean
msoEncodingCyrillic
msoEncodingCyrillicAutoDetect
msoEncodingEBCDICArabic
msoEncodingEBCDICDenmarkNorway
msoEncodingEBCDICFinlandSweden
msoEncodingEBCDICFrance
msoEncodingEBCDICGermany
msoEncodingEBCDICGreek
msoEncodingEBCDICGreekModern
msoEncodingEBCDICHebrew
msoEncodingEBCDICIcelandic
msoEncodingEBCDICInternational
msoEncodingEBCDICItaly
msoEncodingEBCDICJapaneseKatakanaExtended
msoEncodingEBCDICKatakanaExtendedAndJapanese
msoEncodingEBCDICKoreanExtended
msoEncodingEBCDICKoreanExtendedAndKorean
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICMultilingualROECELatin2

msoEncodingEBCDICRussian
msoEncodingEBCDICSerbianBulgarian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICThai
msoEncodingEBCDICTurkish
msoEncodingEBCDICTurkishLatin5
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanada
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEBCDICUSCanadaAndTraditionalChinese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
msoEncodingISCIIAssamese
msoEncodingISCIIBengali
msoEncodingISCIIDevanagari
msoEncodingISCIIGujarati
msoEncodingISCIIKannada
msoEncodingISCIIMalayalam
msoEncodingISCIIOriya
msoEncodingISCIIPunjabi
msoEncodingISCIITamil

msoEncodingISCIITelugu
msoEncodingISO2022CNSimplifiedChinese
msoEncodingISO2022CNTraditionalChinese
msoEncodingISO2022JPJISX02011989
msoEncodingISO2022JPJISX02021984
msoEncodingISO2022JPNoHalfwidthKatakana
msoEncodingISO2022KR
msoEncodingISO6937NonSpacingAccent
msoEncodingISO885915Latin9
msoEncodingISO88591Latin1
msoEncodingISO88592CentralEurope
msoEncodingISO88593Latin3
msoEncodingISO88594Baltic
msoEncodingISO88595Cyrillic
msoEncodingISO88596Arabic
msoEncodingISO88597Greek
msoEncodingISO88598Hebrew
msoEncodingISO88599Turkish
msoEncodingJapaneseAutoDetect
msoEncodingJapaneseShiftJIS
msoEncodingKOI8R
msoEncodingKOI8U
msoEncodingKorean
msoEncodingKoreanAutoDetect
msoEncodingKoreanJohab
msoEncodingMacArabic
msoEncodingMacCroatia
msoEncodingMacCyrillic
msoEncodingMacGreek1
msoEncodingMacHebrew
msoEncodingMacIcelandic
msoEncodingMacJapanese
msoEncodingMacKorean

msoEncodingMacLatin2
msoEncodingMacRoman
msoEncodingMacRomania
msoEncodingMacSimplifiedChineseGB2312
msoEncodingMacTraditionalChineseBig5
msoEncodingMacTurkish
msoEncodingMacUkraine
msoEncodingOEMArabic
msoEncodingOEMBaltic
msoEncodingOEMCanadianFrench
msoEncodingOEMCyrillic
msoEncodingOEMCyrillicII
msoEncodingOEMGreek437G
msoEncodingOEMHebrew
msoEncodingOEMIcelandic
msoEncodingOEMModernGreek
msoEncodingOEMMultilingualLatinI
msoEncodingOEMMultilingualLatinII
msoEncodingOEMNordic
msoEncodingOEMPortuguese
msoEncodingOEMTurkish
msoEncodingOEMUnitedStates
msoEncodingSimplifiedChineseAutoDetect
msoEncodingSimplifiedChineseGBK
msoEncodingT61
msoEncodingTaiwanCNS
msoEncodingTaiwanEten
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTCA
msoEncodingTaiwanTeleText
msoEncodingTaiwanWang
msoEncodingThai
msoEncodingTraditionalChineseAutoDetect

msoEncodingTraditionalChineseBig5
msoEncodingTurkish
msoEncodingUnicodeBigEndian
msoEncodingUnicodeLittleEndian
msoEncodingUSASCII
msoEncodingUTF7
msoEncodingUTF8
msoEncodingVietnamese
msoEncodingWestern

expression.**Encoding**

expression Required. An expression that returns a **WebOptions** object.

Remarks

If the **AlwaysSaveInDefaultEncoding** property is set to **True** on a given **WebOptions** object, any subsequent attempts to set the **Encoding** property on that object will be ignored.

Attempting to set the **Encoding** property to an **MsoEncoding** constant that is not available on the client computer results in a run-time error.

Example

The following example tests whether the Web publication is currently set to be saved using default encoding. If so, the **AlwaysSaveInDefaultEncoding** property is set to **False**, and the **Encoding** property is used to set the encoding to Unicode (UTF-8).

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    If .AlwaysSaveInDefaultEncoding = True Then
        .AlwaysSaveInDefaultEncoding = False
        .Encoding = msoEncodingUTF8
    End If
End With
```

End Property

Sets or returns a **Long** that represents the ending character position of a selection or text range. Read/write.

expression.**End**

expression Required. An expression that returns a **TextRange** object.

Example

This example starts the selection on the 50th character of the current text box shape and ends on the 150th character, then bolds the text.

```
Sub test2()
    With Selection.TextRange
        .Start = 50
        .End = 150
        .Font.Bold = msoTrue
    End With
End Sub
```


Show All

EndArrowheadLength Property

Returns or sets an **MsoArrowheadLength** indicating the length of the arrowhead at the end of the specified line. Read/write.

MsoArrowheadLength can be one of these MsoArrowheadLength constants.

msoArrowheadLengthMedium

msoArrowheadLengthMixed Return value only; indicates a combination of the other states in the specified shape range.

msoArrowheadLong

msoArrowheadShort

expression.**EndArrowheadLength**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **BeginArrowheadLength** property to return or set the length of the arrowhead at the beginning of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

EndArrowheadStyle Property

Returns or sets an **MsoArrowheadStyle** constant indicating the style of the arrowhead at the end of the specified line. Read/write.

MsoArrowheadStyle can be one of these MsoArrowheadStyle constants.

msoArrowheadDiamond

msoArrowheadNone

msoArrowheadOpen

msoArrowheadOval

msoArrowheadStealth

msoArrowheadStyleMixed Return value only; indicates a combination of the other states in the specified shape range.

msoArrowheadTriangle

expression.**EndArrowheadStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **BeginArrowheadStyle** property to return or set the style of the arrowhead at the beginning of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

EndArrowheadWidth Property

Returns or sets an **MsoArrowheadWidth** constant indicating the width of the arrowhead at the end of the specified line. Read/write.

MsoArrowheadWidth can be one of these MsoArrowheadWidth constants.

msoArrowheadNarrow

msoArrowheadWide

msoArrowheadWidthMedium

msoArrowheadWidthMixed Return value only; indicates a combination of the other states in the specified shape range.

expression.**EndArrowheadWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **BeginArrowheadWidth** property to return or set the width of the arrowhead at the beginning of the line.

Example

This example adds a line to the active publication. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=100, BeginY:=100, _
        EndX:=200, EndY:=300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```


Show All

EndConnected Property

Returns an **MsoTriState** constant indicating whether the end of the specified connector is connected to a shape. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The end of the specified connector is not connected to a shape.

msoTriStateMixed Return value only; indicates a combination of **msoTrue** and **msoFalse** in the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The end of the specified connector is connected to a shape.

expression.**EndConnected**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **BeginConnected** property to determine if the beginning of a connector is connected to a shape.

Example

If the third shape on the first page in the active publication is a connector whose end is connected to a shape, this example stores the connection site number, stores a reference to the connected shape, and then disconnects the end of the connector from the shape.

```
Dim intSite As Integer
Dim shpConnected As Shape

With ActiveDocument.Pages(1).Shapes(3)

    ' Test whether shape is a connector.
    If .Connector Then
        With .ConnectorFormat

            ' Test whether connector is connected to another shape.
            If .End Connected Then

                ' Store connection site number.
                intSite = .EndConnectionSite

                ' Set reference to connected shape.
                Set shpConnected = .EndConnectedShape

                ' Disconnect connector and shape.
                .EndDisconnect
            End If
        End With
    End If
End With
```

EndConnectedShape Property

Returns a **Shape** object that represents the shape to which the end of the specified connector is attached.

expression.**EndConnectedShape**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the end of the specified connector isn't attached to a shape, an error occurs.

Use the **BeginConnectedShape** property to return the shape attached to the beginning of a connector.

Example

This example assumes that the first page in the active publication already contains two shapes attached by a connector named Conn1To2. The code adds a rectangle and a connector to the first page. The end of the new connector will be attached to the same connection site as the end of the connector named Conn1To2, and the beginning of the new connector will be attached to connection site one on the new rectangle.

```
Dim shpNew As Shape
Dim intSite As Integer
Dim shpOld As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add new rectangle.
    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=450, Top:=190, Width:=200, Height:=100)

    ' Add new connector.
    .AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=10, EndY:=10) _
        .Name = "Conn1To3"

    ' Get connection site number of old shape, and set
    ' reference to old shape.
    With .Item("Conn1To2").ConnectorFormat
        intSite = .EndConnectionSite
        Set shpOld = .EndConnectedShape
    End With

    ' Connect new connector to old shape and new rectangle.
    With .Item("Conn1To3").ConnectorFormat
        .EndConnect ConnectedShape:=shpOld, _
            ConnectionSite:=intSite
        .BeginConnect ConnectedShape:=shpNew, _
            ConnectionSite:=1
    End With
End With
```

EndConnectionSite Property

Returns a **Long** indicating the connection site to which the end of a connector is connected. Read-only.

expression.**EndConnectionSite**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the end of the specified connector isn't attached to a shape, this property generates an error.

Use the **BeginConnectionSite** property to return the site to which the beginning of a connector is connected.

Example

This example assumes that the first page in the active publication already contains two shapes attached by a connector named Conn1To2. The code adds a rectangle and a connector to the first page. The end of the new connector will be attached to the same connection site as the end of the connector named Conn1To2, and the beginning of the new connector will be attached to connection site one on the new rectangle.

```
Dim shpNew As Shape
Dim intSite As Integer
Dim shpOld As Shape

With ActiveDocument.Pages(1).Shapes

    ' Add new rectangle.
    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=450, Top:=190, Width:=200, Height:=100)

    ' Add new connector.
    .AddConnector(Type:=msoConnectorCurve, _
        BeginX:=0, BeginY:=0, EndX:=10, EndY:=10) _
        .Name = "Conn1To3"

    ' Get connection site number of old shape, and set
    ' reference to old shape.
    With .Item("Conn1To2").ConnectorFormat
        intSite = .EndConnectionSite
        Set shpOld = .EndConnectedShape
    End With

    ' Connect new connector to old shape and new rectangle.
    With .Item("Conn1To3").ConnectorFormat
        .EndConnect ConnectedShape:=shpOld, _
            ConnectionSite:=intSite
        .BeginConnect ConnectedShape:=shpNew, _
            ConnectionSite:=1
    End With
End With
```


Show All

Engrave Property

Returns or sets an **MsoTriState** constant indicating whether the specified text is formatted as engraved. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse The specified text is not formatted as engraved.

msoTriStateMixed Return value which indicates a combination of **msoTrue** and **msoFalse** for the specified text range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue The specified text is formatted as engraved.

expression.**Engrave**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Setting **Engrave** to **msoTrue** sets **Emboss** to **msoFalse**, and vice versa.

Example

This example engraves all the fonts in the first story.

```
Sub FontEng()

    Dim fntEng As Font

    Set fntEng = Application.ActiveDocument. _
        Stories(1).TextRange.Font

    If fntEng.Engrave = msoFalse Or msoTriStateMixed Then
        fntEng.Engrave = msoTrue
    End If

End Sub
```


Show All

EnvelopePrintOrientation Property

Returns or sets a **PbOrientationType** constant that represents the orientation used to print envelopes. Read/write.

PbOrientationType can be one of these PbOrientationType constants.
pbOrientationLandscape
pbOrientationPortrait

expression.**EnvelopePrintOrientation**

expression Required. An expression that returns one of the objects in the Applies To list.

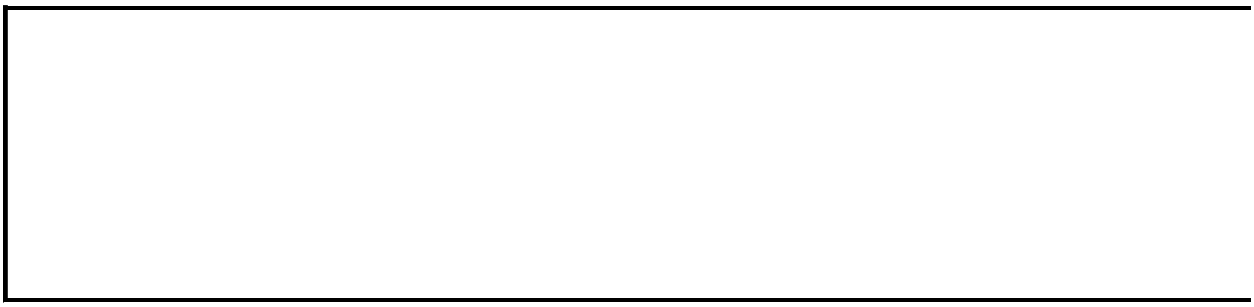
Remarks

Returns 'Permission Denied' for publications that are not envelopes.

Example

This example sets envelope printing options, including the orientation of envelopes. This example assumes the publication is an envelope.

```
Sub SetEnvelopeOptions()
    With Options
        .UseEnvelopePrintOptions = True
        .UseEnvelopePaperSizes = True
        .EnvelopePrintOrientation = pbOrientationLandscape
        .EnvelopePrintPlacement = pbPlacementLeft
    End With
End Sub
```




Show All

EnvelopePrintPlacement Property

Returns or sets a **PbPlacementType** constant that represents the placement of envelopes in the printer tray. Read/write.

PbPlacementType can be one of these PbPlacementType constants.

pbPlacementCenter

pbPlacementLeft

pbPlacementRight

expression.**EnvelopePrintPlacement**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Returns 'Permission Denied' for publications that are not envelopes.

Example

This example sets envelope printing options, including specifying how envelopes are placed in the printer tray. This example assumes the publication is an envelope.

```
Sub SetEnvelopeOptions()
    With Options
        .UseEnvelopePrintOptions = True
        .UseEnvelopePaperSizes = True
        .EnvelopePrintOrientation = pbOrientationLandscape
        .EnvelopePrintPlacement = pbPlacementLeft
    End With
End Sub
```

EnvelopeVisible Property

Returns or sets a **Boolean** indicating whether the e-mail message header is visible in the publication window. Read/write.

expression.**EnvelopeVisible**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the e-mail message header for the active publication.

```
ActiveDocument.EnvelopeVisible = True
```


Show All

Exists Property

 As it applies to the **BorderArtFormat** object.

True if the specified **BorderArtFormat** object exists. Read-only **Boolean**.

expression.**Exists**()

expression Required. An expression that returns a **BorderArtFormat** object.

 As it applies to the **PageBackground** object.

True if the specified **PageBackground** object exists. Read/write **Boolean**.

expression.**Exists**

expression Required. An expression that returns a **PageBackground** object.

Example

As it applies to the **BorderArtFormat** object.

The following example tests for the existence of BorderArt on each shape for each page of the active publication. If BorderArt exists, it is deleted.

```
Sub DeleteBorderArt()

Dim anyPage As Page
Dim anyShape As Shape

For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .Delete
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```

As it applies to the **PageBackground** object.

The following example tests for the existence of a background on the first page of the active document. If a background does not exist, one is created.

```
If ActiveDocument.Pages(1).Background.Exists = False Then
    ActiveDocument.Pages(1).Background.Create
End If
```


Show All

ExpandUsingKashida Property

Returns or sets an **MsoTriState** constant indicating whether to apply kashida rules while applying tracking to Arabic text. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse Microsoft Publisher does not apply kashida rules while applying tracking to Arabic text.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Set value that toggles the property value between **msoTrue** and **msoFalse**.

msoTrue Publisher does apply kashida rules while applying tracking to Arabic text.

expression.**ExpandUsingKashida**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets Microsoft Publisher to apply kashida rules while applying tracking to Arabic text for all text ranges on page one of the active publication.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    If shpLoop.HasTextFrame Then
        shpLoop.TextFrame.TextRange _
            .Font.ExpandUsingKashida = msoTrue
    End If
Next shpLoop
```

ExtrusionColor Property

Returns a **ColorFormat** object representing the color of the shape's extrusion.

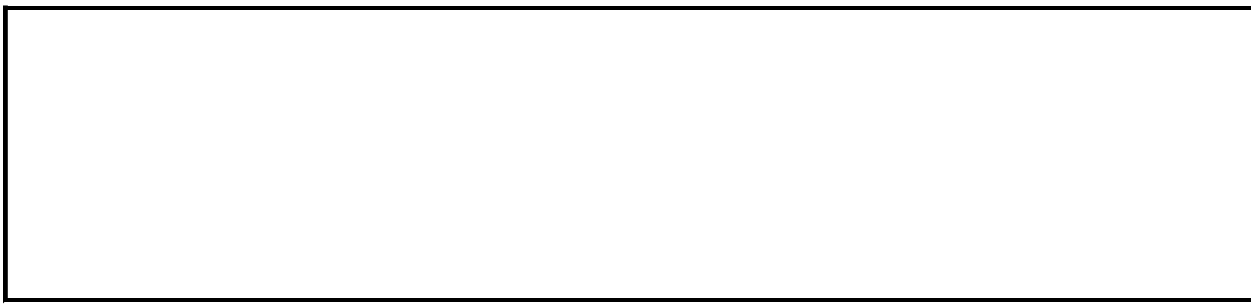
expression.**ExtrusionColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the active publication, and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Dim shpNew As Shape

' Set a reference to a new oval.
Set shpNew = ActiveDocument.Pages(1).Shapes _
    .AddShape(Type:=msoShapeOval, _
    Left:=90, Top:=90, Width:=90, Height:=40)

' Format the 3-D properties of the oval.
With shpNew.ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)
End With
```




Show All

ExtrusionColorType Property

Returns or sets an **MsoExtrusionColorType** constant indicating whether the extrusion color is based on the extruded shape's fill (the front face of the extrusion) and automatically changes when the shape's fill changes, or whether the extrusion color is independent of the shape's fill. Read/write.

MsoExtrusionColorType can be one of these MsoExtrusionColorType constants.

msoExtrusionColorAutomatic Extrusion color is based on shape fill.

msoExtrusionColorCustom Extrusion color is independent of shape fill.

msoExtrusionColorTypeMixed Return value only; indicates a combination of the other states in the specified shape range.

expression.**ExtrusionColorType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

If the first shape in the active publication has an automatic extrusion color, this example gives the extrusion a custom yellow color.

```
With ActiveDocument.Pages(1).Shapes(1).ThreeD
    If .ExtrusionColorType = msoExtrusionColorAutomatic Then
        .ExtrusionColor.RGB = RGB(240, 235, 16)
    End If
End With
```

Fields Property

Returns a **Fields** object that represents all the fields in the specified text range.

expression.**Fields**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example bolds the first field in the first shape on the first page of the active publication.

```
Sub CountFields()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Fields(1).TextRange.Font.Bold = msoTrue
End Sub
```


Show All

FieldType Property

Returns a **pbMailMergeDataFieldType** constant that represents the type of data contained in the data field.

PbMailMergeDataFieldType can be one of these pbMailMergeDataFieldType constants.

pbMailMergeDataFieldPicture

pbMailMergeDataFieldString

expression.**FieldType**

expression Required. An expression that returns a **MailMergeDataField** object.

Remarks

Use the **Insert** method of the **MailMergeDataFields** collection to add a picture data field to a publication's catalog merge area.

Use the **InsertMailMergeField** method of the **TextRange** object to add a text data field to a text box in the publication's catalog merge area.

Example

This example defines a data field as a picture data field, inserts it into the catalog merge area of the specified publication, and sizes and positions the picture data field. This example assumes that the publication has been connected to a data source, and that a catalog merge area has been added to the publication.

```
Dim pbPictureField1 As Shape

    'Define the Photo field as a picture data type
    With ThisDocument.MailMerge.DataSource.DataFields
        .Item("Photo:").FieldType = pbMailMergeDataFieldPicture
    End With

    'Insert a picture field, then size and position it
    Set pbPictureField1 = ThisDocument.MailMerge.DataSource.DataFiel
        With pbPictureField1
            .Height = 100
            .Width = 100
            .Top = 85
            .Left = 375
        End With
```


Show All

FileDialog Property

Returns a **FileDialog** object which represents a single instance of a file dialog box.

expression.**FileDialog**(***Type***)

expression Required. An expression that returns one of the objects in the Applies To list.

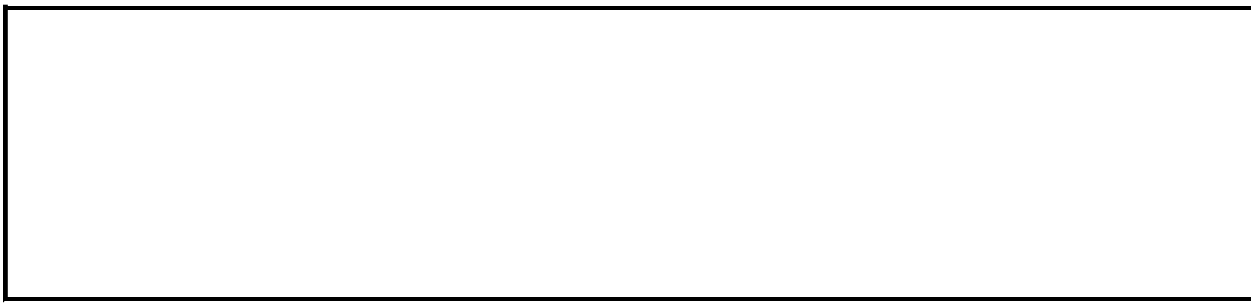
Type Required **MsoFileDialogType**. The type of dialog.

MsoFileDialogType can be one of these MsoFileDialogType constants.
msoFileDialogFilePicker
msoFileDialogFolderPicker
msoFileDialogOpen
msoFileDialogSaveAs

Example

This example displays the **Save As** dialog box and stores the file name specified by the user.

```
Sub ShowSaveAsDialog()
    Dim dlgSaveAs As FileDialog
    Dim strFile As String

    Set dlgSaveAs = Application.FileDialog( _
        Type:=msoFileDialogSaveAs)
    dlgSaveAs.Show
    strFile = dlgSaveAs.SelectedItems(1)
End Sub
```



Filename Property

Returns a **String** that represents the file name of the specified picture or OLE object. Read-only.

expression.**Filename**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

For linked pictures and OLE objects, the returned string represents the full path and file name of the picture. For embedded pictures and OLE objects, the returned string represents the file name only.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Example

The following example returns selected image properties for each picture in the active publication.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPictur

            With shpLoop.PictureFormat

                  If .IsEmpty = msoFalse Then

                       Debug.Print "File Name: " & .Filename
                       Debug.Print "Horizontal Scaling: " & .Horizo
                       Debug.Print "Vertical Scaling: " & .Vertical
                       Debug.Print "File size in publication: " & .

                  End If

            End With
        End If
    Next shpLoop
Next pgLoop
```

FileSearch Property

Returns a **FileSearch** object that can be used to search for files using either an absolute or relative path.

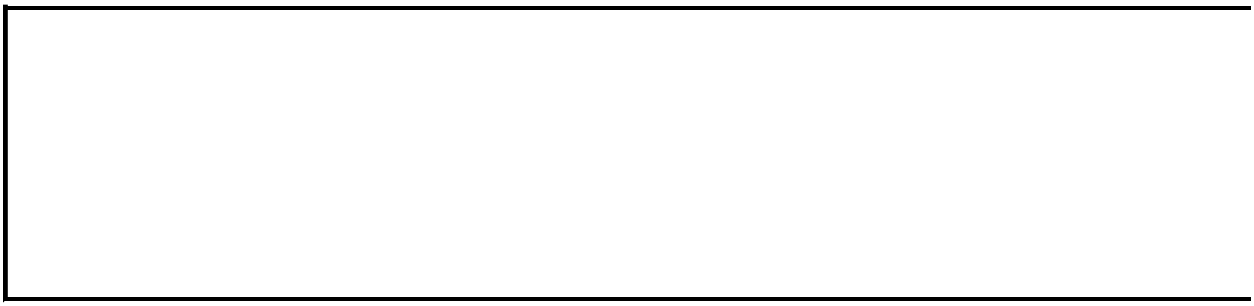
expression.**FileSearch**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays, in a series of message boxes, the file names of all Publisher files in the specified folder. (Note that *PathToFolder* must be replaced with a valid folder path for this example to work.)

```
Sub SearchForFiles()

    Dim intCount As Integer

    With Application.FileSearch
        .FileName = "*.pub"
        .LookIn = "PathToFolder"
        .Execute
        For intCount = 1 To .FoundFiles.Count
            MsgBox .FoundFiles(intCount)
        Next intCount
    End With

End Sub
```



FileSize Property

Returns a **Long** that represents, in bytes, the size of the picture or OLE object as it appears in the specified publication. Read-only.

expression.**FileSize**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

If the picture or OLE object is linked, use the **OriginalFileSize** property to determine the size of the linked file.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Example

The following example tests each picture in the active publication, and prints selected image properties for pictures that are linked.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat

                        Debug.Print "File Name: " & .Filename
                        Debug.Print "Original File Size: " & .Origin
                        Debug.Print "File size in publication: " & .
            End With
        End If
    Next shpLoop
Next pgLoop
```

Fill Property

Returns a **FillFormat** object representing the fill for the specified shape or table cell.

expression.**Fill**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new **AutoShape** object and fills the shape with green.

```
Sub NewShapeItem()

    Dim shpHeart As Shape

    Set shpHeart = ThisDocument.MasterPages.Item(1).Shapes _
        .AddShape(Type:=msoShapeHeart, Left:=40, Top:=80, _
        Width:=50, Height:=50)
    shpHeart.Fill.ForeColor.RGB = RGB(Red:=0, Green:=255, Blue:=0)

End Sub
```


Show All

Filters Property

Returns a **MailMergeFilters** object that represents filters applied to the mail merge or catalog merge data source.

expression.**Filters**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a new filter that removes all records with a blank Region field and then applies the filter to the active publication. This example assumes that a mail merge data source is attached to the active publication.

```
Sub FilterDataSource()
    With ActiveDocument.MailMerge.DataSource
        .Filters.Add Column:="Region", _
            Comparison:=msoFilterComparisonIsBlank, _
            Conjunction:=msoFilterConjunctionAnd
        .ApplyFilter
    End With
End Sub
```


Show All

Find Property

 As it applies to the **Document** object.

Returns a **FindReplace** object. The **FindReplace** object is used to perform a text search and replace in the specified document.

expression.**Find**

expression Required. An expression that returns a **Document** object.

 As it applies to the **TextRange** object.

Returns a **FindReplace** object from the specified **TextRange** object. The **FindReplace** object is used to perform a text search and replace in the specified text range.

expression.**Find**

expression Required. An expression that returns a **TextRange** object.

Example

As it applies to the **Document** object.

The following example sets an object variable to the **FindReplace** object of the active document. A search operation is executed that applies bold formatting to every occurrence of the word "important".

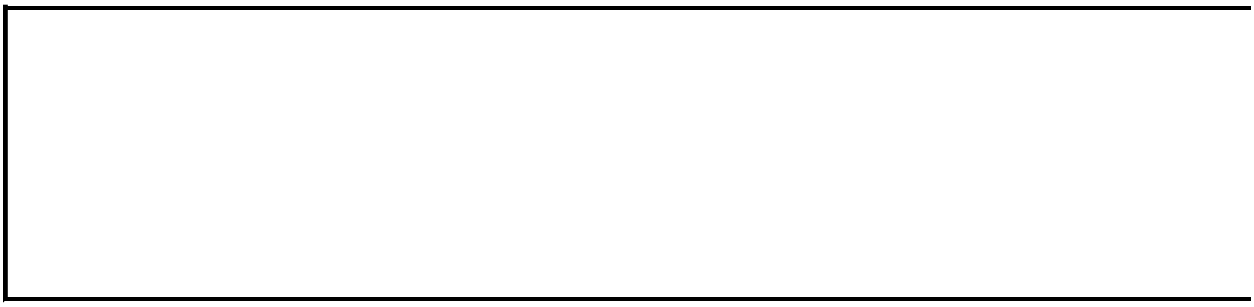
```
        Dim objFind as FindReplace
Dim fFound as Boolean

Set objFind = ActiveDocument.Find
fFound = True

With objFind
    .Clear
    .FindText = "important"
    Do While fFound = True
        fFound = .Execute
        If Not .FoundTextRange Is Nothing Then
            .FoundTextRange.Font.Bold = True
        End If
    Loop
End With
```

As it applies to the **TextRange** object.

The following example sets an object variable to the **FindReplace** object of the text range of the first shape in the active document. A search operation is executed that applies bold formatting to every occurrence of the word "urgent" in the text range.

```
        Dim objFind as FindReplace
Dim fFound as Boolean

Set objFind = ActiveDocument.Pages(1) _
    .Shapes(1).TextFrame.TextRange.Find
fFound = True

With objFind
    .Clear
    .FindText = "urgent"
    Do While fFound = True
```

```
        fFound = .Execute
        If Not .FoundTextRange Is Nothing Then
            .FoundTextRange.Font.Bold = True
        End If
    Loop
End With
```



FindText Property

Sets or retrieves a **String** representing the text to find in the specified range or selection. Read/write.

expression.**FindText**

expression Required. An expression that returns a **FindReplace** object.

Remarks

The **FindText** property returns the plain, unformatted text of the selection. When you set this property, the search text is specified. You can search for special characters by specifying appropriate character codes. For example, "^p" corresponds to a paragraph mark and "^t" corresponds to a tab character.

The default value for the **FindText** property is an empty string. Because only text searching is supported, **FindText** must be explicitly set to avoid a runtime error.

Examples

This example replaces all occurrences of the word "This" in the selection with "That" in each open publication.

```
Dim objDocument As Document

For Each objDocument In Documents
    With objDocument.Find
        .Clear
        .MatchCase = True
        .FindText = "This"
        .ReplaceWithText = "That"
        .ReplaceScope = pbReplaceScopeAll
        .Forward = True
        .Execute
    End With
Next objDocument
```

This example replaces all tab characters with paragraph marks.

```
Dim objDocument As Document

For Each objDocument In Documents
    With objDocument.Find
        .Clear
        .MatchCase = True
        .FindText = "^t"
        .ReplaceWithText = "^p"
        .ReplaceScope = pbReplaceScopeAll
        .Execute
    End With
Next objDocument
```

FirstLineIndent Property

Returns or sets a **Variant** that represents the amount of space (measured in points) to indent the first line in a paragraph. Read/write.

expression.**FirstLineIndent**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a text box, fills it with text, and indents the first line of every paragraph a half inch.

```
Sub IndentFirstLines()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes _
            .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
            Left:=100, Top:=100, Width:=100, Height:=100) _
            .TextFrame.TextRange
        For intCount = 1 To 10
            .InsertAfter NewText:="This is a test. "
        Next intCount
        .ParagraphFormat.FirstLineIndent = InchesToPoints(0.5)
    End With
End Sub
```


Show All

FirstRecord Property

Returns or sets a **Long** that represents the number of the first data record to be merged in a mail merge or catalog merge operation. Read/write.

expression.**FirstRecord**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the active record as the first record to be merged, and then merges three records ending with the record two records forward in the data source. This example assumes that the active publication is a mail merge document.

```
Sub RecordOne()
    With ActiveDocument.MailMerge
        .DataSource.FirstRecord = .DataSource.ActiveRecord
        .DataSource.LastRecord = .DataSource.ActiveRecord + 2
        .Execute Pause:=True
    End With
End Sub
```

Font Property

Sets or returns a **Font** object that represents character formatting attributes applied to the specified object. Read/write.

expression.**Font**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example selects text and formats the font as bold.

```
Sub test2()
    With Selection.TextRange
        .Start = 50
        .End = 150
        .Font.Bold = msoTrue
    End With
End Sub
```


Show All

FontBold Property

Sets or returns an **MsoTriState** constant that represents whether the font for a dropped capital letter or WordArt text effect is bold. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

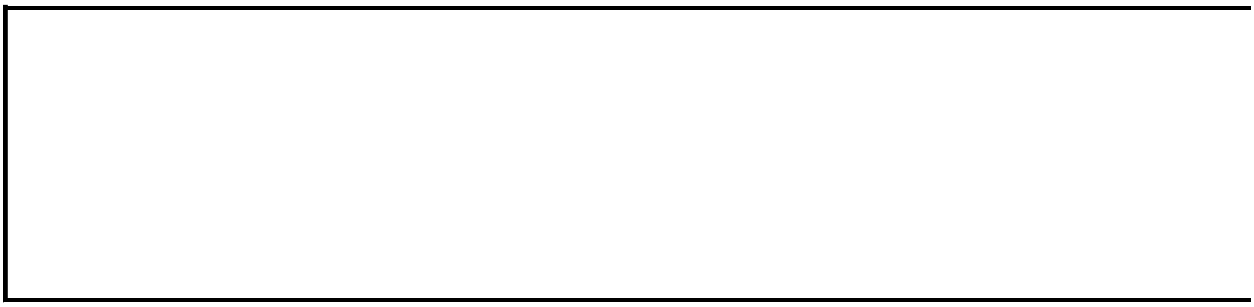
expression.**FontBold**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example applies bold formatting to the dropped capital letter in the specified text frame. This example assumes that the specified text frame is formatted with a dropped capital letter.

```
Sub BoldDropCap()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.DropCap
        .FontBold = msoTrue
        .FontColor.RGB = RGB(Red:=150, Green:=50, Blue:=180)
        .FontItalic = msoTrue
        .FontName = "Script MT Bold"
    End With
End Sub
```



FontColor Property

Returns or sets a **ColorFormat** object that represents the color applied to a specified dropped capital letter.

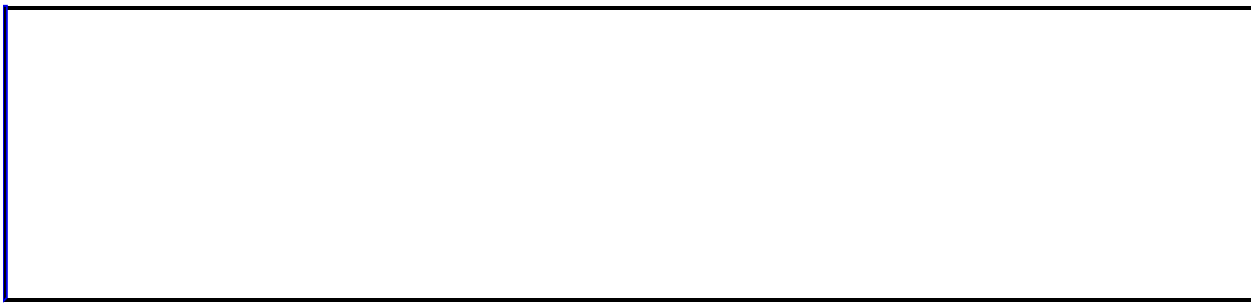
expression.**FontColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example applies an **RGB** color to the dropped capital letter in the specified text frame. This example assumes that the specified text frame is formatted with a dropped capital letter.

```
Sub BoldDropCap()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.DropCap
        .FontBold = msoTrue
        .FontColor.RGB = RGB(Red:=150, Green:=50, Blue:=180)
        .FontItalic = msoTrue
        .FontName = "Script MT Bold"
    End With
End Sub
```




Show All

FontItalic Property

Sets or returns an **MsoTriState** constant that represents whether the font for a dropped capital letter or WordArt text effect is italicized. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

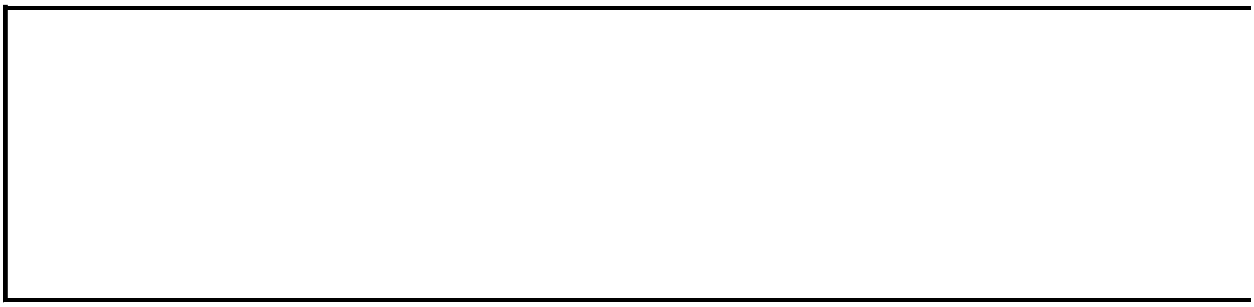
expression.**FontItalic**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example italicizes the dropped capital letter in the specified text frame. This example assumes that the specified text frame is formatted with a dropped capital letter.

```
Sub BoldDropCap()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.DropCap
        .FontBold = msoTrue
        .FontColor.RGB = RGB(Red:=150, Green:=50, Blue:=180)
        .FontItalic = msoTrue
        .FontName = "Script MT Bold"
    End With
End Sub
```




Show All

FontName Property

 FontName property as it applies to the **DropCap** and **TextEffectFormat** objects.

Sets or returns a **String** that represents the name of the font applied to a dropped capital letter or WordArt text effect. Read/write.

expression.**FontName**

expression Required. An expression that returns one of the above objects.

 FontName property as it applies to the **PhoneticGuide** object.

Returns a **String** that represents the name of the font applied to phonetic information displayed above Japanese text. Read-only.

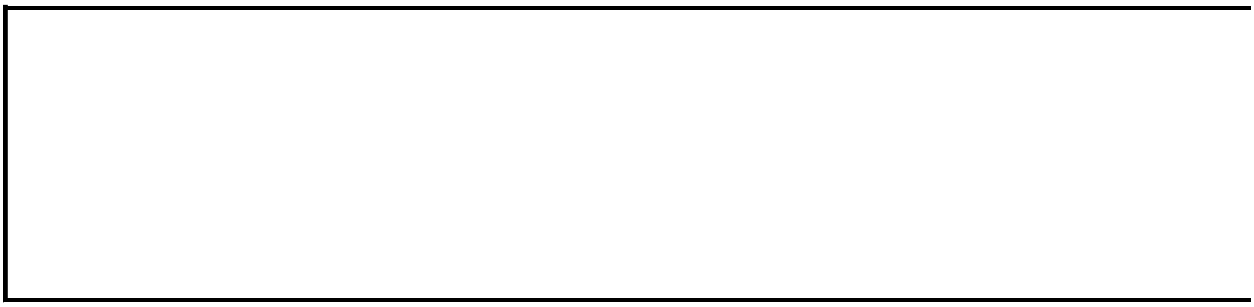
expression.**FontName**

expression Required. An expression that returns one of the above objects.

Example

This example applies the Script MT Bold font to the dropped capital letter in the specified text frame. This example assumes that the specified text frame is formatted with a dropped capital letter.

```
Sub BoldDropCap()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.DropCap
        .FontBold = msoTrue
        .FontColor.RGB = RGB(Red:=150, Green:=50, Blue:=180)
        .FontItalic = msoTrue
        .FontName = "Script MT Bold"
    End With
End Sub
```




Show All

FontSize Property

 FontSize property as it applies to the **TextEffectFormat** object.

Returns or sets a **Variant** that represents the font size for the specified WordArt, in points. Read/write.

expression.**FontSize**

expression Required. An expression that returns one of the above objects.

 FontSize property as it applies to the **PhoneticGuide** object.

Returns a **Variant** that represents the font size of phonetic characters. Read-only.

expression.**FontSize**

expression Required. An expression that returns one of the above objects.

Example

This example sets the font size, name, and bold setting for the specified WordArt shape. This example assume the first shape on the first page of the active publication is a WordArt object.

```
Sub SetWordArtFontSize()
    With ActiveDocument.Pages(1).Shapes(1).TextEffect
        .FontSize = 54
        .FontBold = msoTrue
        .FontName = "Snap ITC"
    End With
End Sub
```

Footer Property

Returns a **HeaderFooter** object representing the footer of the specified **Page** object. Read only.

expression.**Footer**

expression Required. An expression that returns a **Page** object from the **MasterPages** collection.

Remarks

This property is for master pages only. A "This feature is only for master pages" error is returned if the Footer property is accessed from a **Page** object that is returned form the **Pages** collection. A new **HeaderFooter** object is created for the specified master page by accessing this property.

Example

The following example creates a **HeaderFooter** object and sets it to the footer of the first master page.

```
Dim objFooter As HeaderFooter
Set objFooter = ActiveDocument.MasterPages(1).Footer
```

The **HeaderFooter** object returned by the **Footer** property can be used to manipulate the footer content. The following example sets some properties of the **HeaderFooter** object of the first master page,

```
With ActiveDocument.masterPages(1)
    With .Header
        .TextRange.Text = "Windows" & Chr(13) & "Office" & Chr(13) &
        With .TextRange.ParagraphFormat
            .SetListType Value:=pbListTypeBullet, BulletText:="*"
            .Alignment = pbParagraphAlignmentLeft
        End With
    End With
    With .Footer
        .TextRange.Hyperlinks.Add Text:=.TextRange, _
            Address:="http://www.tailspintoys.com", _
            TextToDisplay:="Tailspin"
    End With
End With
```

ForeColor Property

Returns or sets a **ColorFormat** object representing the foreground color for the fill, line, or shadow. Read/write.

expression.**ForeColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **BackColor** property to set the background color for a fill or line.

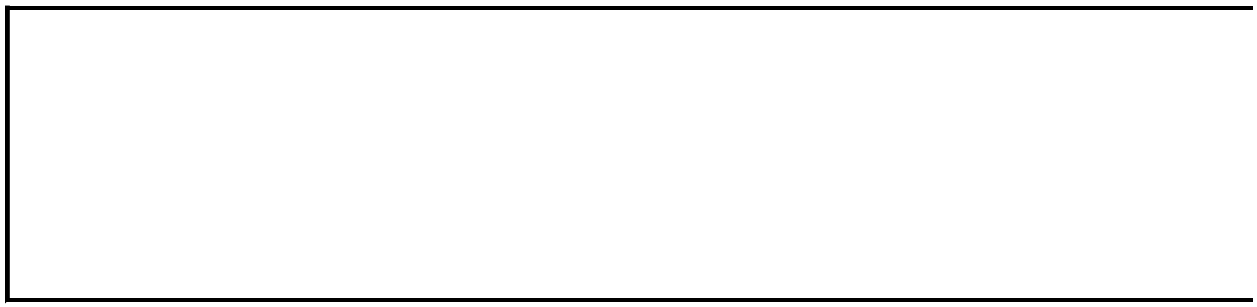
Example

This example adds a rectangle to the active publication and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
With ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeRectangle, _
        Left:=90, Top:=90, Width:=90, Height:=50).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(170, 170, 170)
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

This example adds a patterned line to the active publication.

```
With ActiveDocument.Pages(1).Shapes.AddLine _
        (BeginX:=10, BeginY:=100, EndX:=250, EndY:=0).Line
    .Weight = 6
    .ForeColor.RGB = RGB(0, 0, 255)
    .BackColor.RGB = RGB(128, 0, 0)
    .Pattern = msoPatternDarkDownwardDiagonal
End With
```



Forward Property

Sets or retrieves a **Boolean** representing the direction of the text search. **True** if the find operation searches forward through the document. **False** if it searches backward through the document. Read/write.

expression.**Forward**

expression Required. An expression that returns a **FindReplace** object.

Remarks

Forward must be set to **True** when replacing text.

Example

This example replaces all occurrences of the word "This" in the selection with "That" in each open publication.

```
Dim objDocument As Document

For Each objDocument In Documents
    With objDocument.Find
        .Clear
        .MatchCase = True
        .FindText = "This"
        .ReplaceWithText = "That"
        .ReplaceScope = pbReplaceScopeAll
        .Forward = True
        .Execute
    End With
Next objDocument
```

FoundTextRange Property

Returns a **TextRange** object that represents the found text or replaced text of a find operation. Read-only.

expression.**FoundTextRange**

expression Required. An expression that returns a **FindReplace** object.

Remarks

The actual **TextRange** returned by the **FoundTextRange** property is determined by the value of the **pbReplaceScope** property. The following table lists the corresponding values of these properties.

for	**pbReplaceScopeAll**	**FoundTextRange** = Empty
for	**pbReplaceScopeNone**	**FoundTextRange** = Find text range
for	**pbReplaceScopeOne**	**FoundTextRange** = Replace text range

When **ReplaceScope** is set to **pbReplaceScopeAll** the **FoundTextRange** is empty. Any attempt to access it will return Access denied. The way to manipulate the text range of the searched text is to set the **ReplaceScope** to **pbReplaceScopeNone** or **pbReplaceScopeOne** and access the text range of the searched or replaced text for each occurrence found.

When **ReplaceScope** is set to **pbReplaceScopeNone**, **FoundTextRange** returns the text range of the searched text. The following example illustrates how the font attributes of the find text range can be accessed when **ReplaceScope** is set to **pbReplaceScopeNone**.

```
With TextRange.Find
    .Clear
    .FindText = "important"
    .ReplaceScope = pbReplaceScopeNone
    Do While .Execute = True
        'The FoundTextRange contains the word "important".
        If .FoundTextRange.Font.Italic = msoFalse Then
            .FoundTextRange.Font.Italic = msoTrue
        End If
    Loop
End With
```

When **ReplaceScope** is set to **pbReplaceScopeOne** the text range of the searched text is replaced. Therefore the **FoundTextRange** returns the text range of the replacement text. The following example demonstrates how the font attributes of the replaced text range can be accessed when **ReplaceScope** is set to **pbReplaceScopeOne**.

```
With Document.Find
```

```
    .Clear
    .FindText = "important"
    .ReplaceWithText = "urgent"
    .ReplaceScope = pbReplaceScopeOne
    Do While .Execute = True
        'The FoundTextRange contains the word "urgent".
        If .FoundTextRange.Font.Bold = msoFalse Then
            .FoundTextRange.Font.Bold = msoTrue
        End If
    Loop
End With
```

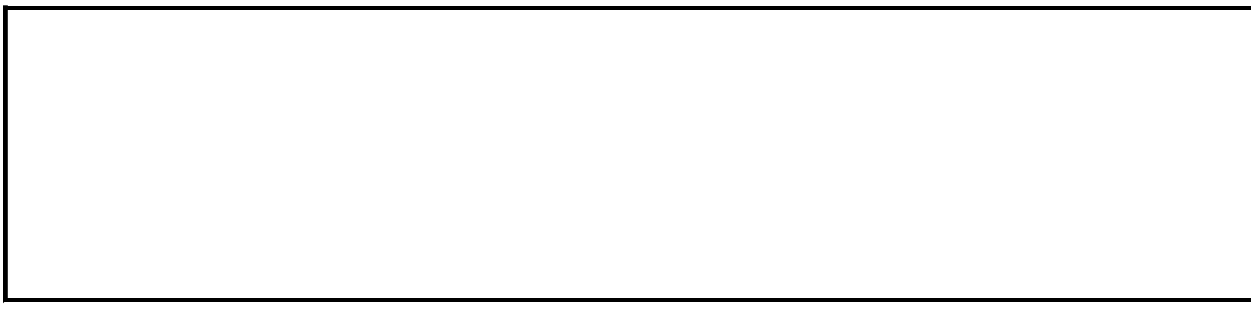
Examples

This example replaces each example of the word "bizarre" with the word "strange" and applies italics and bold formatting to the replaced text.

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "bizarre"
    .ReplaceWithText = "strange"
    .ReplaceScope = pbReplaceScopeOne
    Do While .Execute = True
        .FoundTextRange.Font.Italic = msoTrue
        .FoundTextRange.Font.Bold = msoTrue
    Loop
End With
```

This example finds all occurrences of the word "important" and applies italics formatting to it.

```
Dim objTextRange As TextRange

Set objTextRange = ActiveDocument.Pages(1).Shapes(1).TextFrame.TextR
With objTextRange.Find
    .Clear
    .FindText = "important"
    .ReplaceScope = pbReplaceScopeNone
    Do While .Execute = True
        .FoundTextRange.Font.Italic = msoTrue
    Loop
End With
```



Frequency Property

Returns or sets a **Long** indicating the number of lines per inch that the plate will print. The default is 133. Read/write.

expression.**Frequency**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

To specify a custom frequency setting for a printable plate, the **UseCustomHalftone** of the **AdvancedPrintOptions** object must be set to **True**. Returns "Permission Denied" if the **UseCustomHalftone** is set to **False**.

Example

This example sets the spot color plates (plates five and higher) of a process and spot color publication to the same custom angle and frequency. The example assumes that the publication's color mode has been specified as process and spot colors, and the publication's print mode has been specified as separations.

```
Sub SetSpotColorPlatesProperties()

ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

Dim intCount As Integer

With ActiveDocument.AdvancedPrintOptions.PrintablePlates
    For intCount = 5 To .Count
        With .Item(intCount)
            .Angle = 45
            .Frequency = 150
        End With
    Next
End With

End Sub
```

FullName Property

Returns a **String** representing the full file name of the saved active publication, including its path and file name. Read-only.

expression.**FullName**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **FullName** property can be used to return both path and file name as returned by the **Path** and **Name** properties.

Example

The following example demonstrates the differences between the **Path**, **Name**, and **FullName** properties. This example is best illustrated if the publication is saved in a folder other than the default.

```
Sub PathNames()

    Dim strPath As String
    Dim strName As String
    Dim strFullName As String

    strPath = Application.ActiveDocument.Path
    strName = Application.ActiveDocument.Name
    strFullName = Application.ActiveDocument.FullName

    ' Note the file name & path differences
    ' while executing.
    MsgBox "The path is: " & strPath
    MsgBox "The file name is: " & strName
    MsgBox "The path & file name are: " & strFullName

End Sub
```

Gap Property

Returns or sets a **Variant** indicating the horizontal distance between the end of the callout line and the text bounding box. Read/write.

expression.**Gap**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

Example

This example sets the distance between the callout line and the text bounding box to 3 points for the first shape in the active publication. For the example to work, the shape must be a callout.

```
ActiveDocument.Pages(1).Shapes(1).Callout.Gap = 3
```


Show All

GradientColorType Property

Returns an **MsoGradientColorType** constant indicating the gradient color type for the specified fill. Read-only.

MsoGradientColorType can be one of these MsoGradientColorType constants.

msoGradientColorMixed Return value only; indicates a combination of the other states in the specified range.

msoGradientOneColor

msoGradientPresetColors

msoGradientTwoColors

expression.**GradientColorType**

expression Required. An expression that returns one of the objects in the Applies To list.

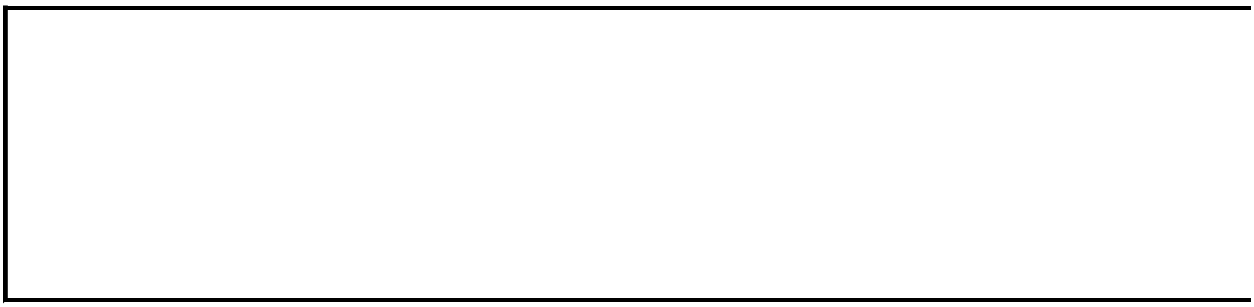
Remarks

Use the **OneColorGradient**, **PresetGradient**, or **TwoColorGradient** method to set the gradient type for the fill.

Example

This example changes the fill for all shapes on the first page of the active publication that have a two-color gradient fill to a preset gradient fill.

```
Dim shpLoop As Shape

' Loop through collection of shapes.
For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop.Fill
        ' Test for two-color gradient.
        If .GradientColorType = msoGradientTwoColors Then
            ' Apply a preset gradient.
            .PresetGradient Style:=msoGradientHorizontal, _
                Variant:=1, PresetGradientType:=msoGradientBrass
        End If
    End With
Next shpLoop
```



GradientDegree Property

Returns a **Single** indicating how dark or light a one-color gradient fill is. A value of 0 (zero) means that black is mixed in with the shape's foreground color to form the gradient; a value of 1 means that white is mixed in; and values between 0 and 1 mean that a darker or lighter shade of the foreground color is mixed in. Read-only.

expression.**GradientDegree**

expression Required. An expression that returns one of the objects in the Applies To list.

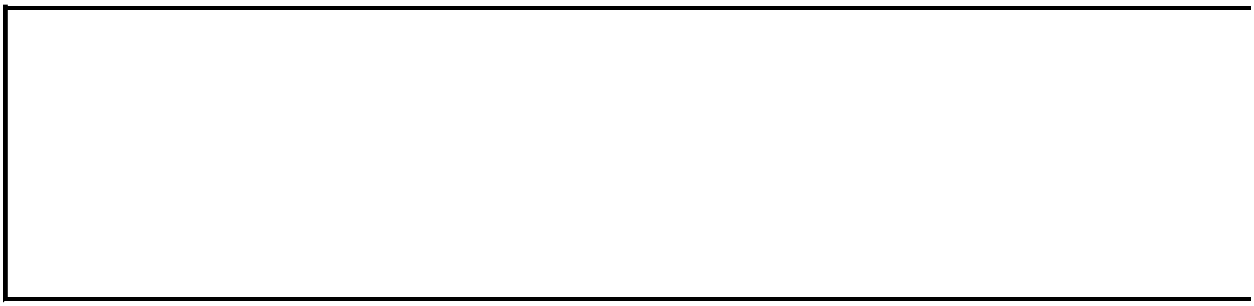
Remarks

Use the **OneColorGradient** method to set the gradient degree for the fill.

Example

This example adds a rectangle to the active publication and sets the degree of its fill gradient to match that of the shape named Rectangle 2. If Rectangle 2 doesn't have a one-color gradient fill, this example generates an error.

```
Dim sngDegree As Single

With ActiveDocument.Pages(1).Shapes
    ' Store degree of one-color gradient.
    sngDegree = .Item("Rectangle 2").Fill.GradientDegree
    ' Add new rectangle.
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        ' Set color and gradient for new rectangle.
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient Style:=msoGradientHorizontal, _
            Variant:=1, Degree:=sngDegree
    End With
End With
```




Show All

GradientStyle Property

Returns an **MsoGradientStyle** constant indicating the gradient style for the specified fill. Read-only.

MsoGradientStyle can be one of these MsoGradientStyle constants.

msoGradientDiagonalDown

msoGradientDiagonalUp

msoGradientFromCenter

msoGradientFromCorner

msoGradientFromTitle

msoGradientHorizontal

msoGradientMixed Return value only; indicates a combination of the other states in the specified shape range.

msoGradientVertical

expression.**GradientStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **OneColorGradient**, **PresetGradient**, or **TwoColorGradient** method to set the gradient style for the fill.

Attempting to return this property for a fill that doesn't have a gradient generates an error. Use the **Type** property to determine whether the fill has a gradient.

Example

This example adds a rectangle to the active publication and sets its fill gradient style to match that of the shape named rect1. For the example to work, rect1 must have a gradient fill.

```
Dim intStyle As Integer

With ActiveDocument.Pages(1).Shapes
    ' Store gradient style of rect1.
    intStyle = .Item("rect1").Fill.GradientStyle
    ' Add new rectangle.
    With .AddShape(Type:=msoShapeRectangle, _
            Left:=0, Top:=0, Width:=40, Height:=80).Fill
        ' Set color and gradient of new rectangle.
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient Style:=intStyle, _
            Variant:=1, Degree:=1
    End With
End With
```

GradientVariant Property

Returns a **Long** indicating the gradient variant for the specified fill. Generally, values are integers from 1 to 4 for most gradient fills. If the gradient style is **msoGradientFromTitle** or **msoGradientFromCenter**, this property returns either 1 or 2. The values for this property correspond to the gradient variants (numbered from left to right and from top to bottom) on the **Gradient** tab in the **Fill Effects** dialog box. Read-only.

expression.**GradientVariant**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **OneColorGradient**, **PresetGradient**, or **TwoColorGradient** method to set the gradient variant for the fill.

Example

This example adds a rectangle to the active publication and sets its fill gradient variant to match that of the shape named rect1. For the example to work, rect1 must have a gradient fill.

```
Dim intVariant As Integer

With ActiveDocument.Pages(1).Shapes
    ' Store gradient variant of rect1.
    intVariant = .Item("rect1").Fill.GradientVariant
    ' Add new rectangle.
    With .AddShape(Type:=msoShapeRectangle, _
            Left:=0, Top:=0, Width:=40, Height:=80).Fill
        ' Set color and gradient of new rectangle.
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient Style:=msoGradientHorizontal, _
            Variant:=intVariant, Degree:=1
    End With
End With
```


Show All

GraphicsResolution Property

Returns or sets a **PbPrintGraphics** constant representing the resolution at which the inserted graphics are to be printed in the specified publication. Read/write.

PbPrintGraphics can be one of these PbPrintGraphics constants.

pbPrintHighResolution *Default*. Print linked graphics using the full-resolution linked version.

pbPrintLowResolution Print linked graphics using the low-resolution placeholder version that is stored in the publication.

pbPrintGraphicsNoGraphics Print a box in place of linked graphics.

expression.**GraphicsResolution**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Setting this property only affects inserted pictures (whether linked or embedded), and clip art. Autoshapes and border art will always print.

Printing boxes in place of graphics is useful when printing a quick proof of the layout that only shows the positioning of pictures.

This property corresponds to the **Graphics** controls on the **Graphics and Fonts** tab of the **Advanced Print Settings** dialog box.

Example

The following example sets the graphics to print as boxes in the active publication.

```
Sub PrintGraphicAsBoxes
        With ActiveDocument.AdvancedPrintOptions
                If .GraphicsResolution <> pbPrintNoGraphics Then
                .GraphicsResolution = pbPrintNoGraphics
                End If
        End With
End Sub
```

GroupItems Property

Returns a **GroupShapes** collection if the specified shape is a group.

expression.**GroupItems**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

All smart objects will be treated as grouped shapes.

Example

This example adds three triangles to a publication, groups them, sets a color for the entire group, and then changes the color for the second triangle only.

```
Sub Grouper()

    Dim docSheet As Document

    Set docSheet = ActiveDocument
    With docSheet.MasterPages.Item(1).Shapes
        ' Add the 3 triangles
        .AddShape(Type:=msoShapeIsoscelesTriangle, _
            Left:=10, Top:=10, Width:=100, Height:=100).Name = "shpO
        .AddShape(Type:=msoShapeIsoscelesTriangle, _
            Left:=150, Top:=10, Width:=100, Height:=100).Name = "shp
        .AddShape(Type:=msoShapeIsoscelesTriangle, _
            Left:=300, Top:=10, Width:=100, Height:=100).Name = "shp
        ' Group and fill the 3 triangles
        With .Range(Array("shpOne", "shpTwo", "shpThree")).Group
            .Fill.PresetTextured msoTextureBlueTissuePaper
            .GroupItems(2).Fill.PresetTextured msoTextureGreenMarble
        End With
    End With

End Sub
```

GrowToFitText Property

True for cells in a table to increase vertically to fit text. Read/write **Boolean**.

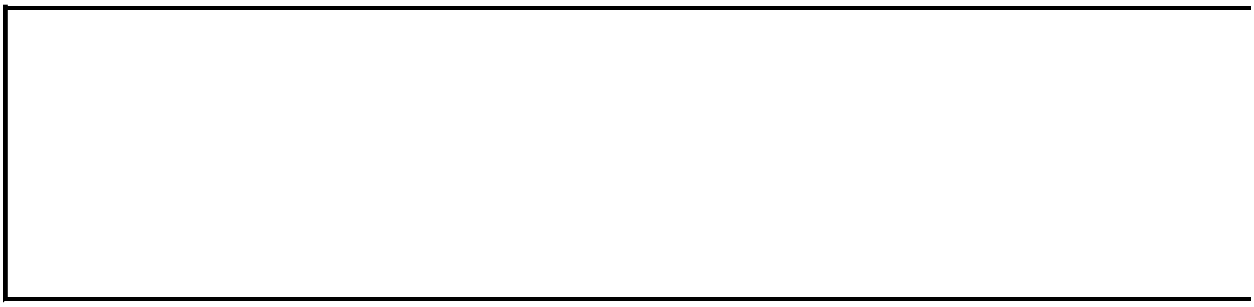
expression.**GrowToFitText**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets each row of the specified table to 12 points, and the row height doesn't increase as text is added to the cells in the rows.

```
Sub DontEnlargeTableCells()
    Dim rowTable As Row
    With ActiveDocument.Pages(1).Shapes(1).Table
        .GrowToFitText = False
        For Each rowTable In .Rows
            rowTable.Height = 12
        Next
    End With
End Sub
```



GutterCenterlines Property

Returns or sets a value that specifies whether to add a center line between the columns and rows of the gutter guides in a master page. Read/write **Boolean**.

expression.**GutterCenterlines**

expression Required. An expression that returns a **LayoutGuides** object.

Remarks

The **GutterCenterlines** property can only be used if the **LayoutGuides.Rows** property or the **LayoutGuides.Columns** property is greater than 1.

If **True**, a red line appears in the center of the gutter guides. If **False**, no line appears in the center of the gutter guides. The default value is **False**.

Example

The following example modifies the first master page of the active publication to have three rows, three columns, and red center lines drawn in the gutter guides. Any pages added to the publication after this point will have red center lines drawn in the gutter guides.

```
Dim theMasterPage As page
Dim theLayoutGuides As LayoutGuides

Set theMasterPage = ActiveDocument.MasterPages(1)
Set theLayoutGuides = theMasterPage.LayoutGuides

With theLayoutGuides
    .Rows = 3
    .Columns = 3
    .GutterCenterlines = True
End With
```


Show All

HasAlphaChannel Property

Returns an **MsoTriState** constant indicating whether the specified picture contains an alpha channel. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The specified picture does not contain an alpha channel.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified picture contains an alpha channel.

expression.**HasAlphaChannel**()

expression Required. An expression that returns a **PictureFormat** object.

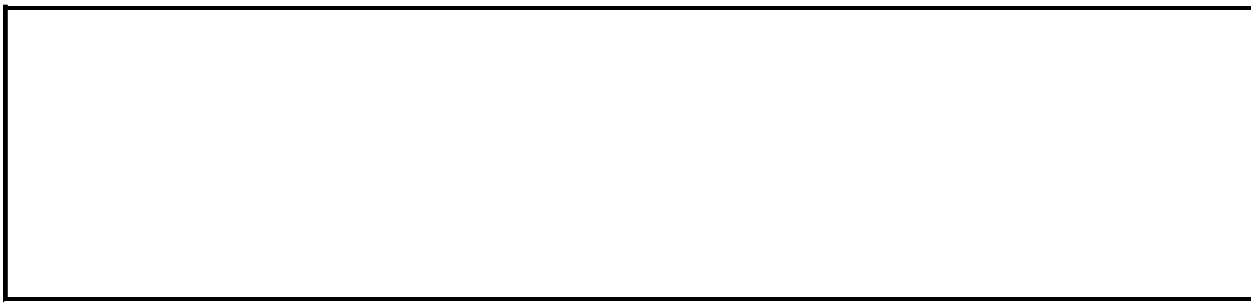
Remarks

An alpha channel is a special 8-bit channel used by some image processing software to contain additional data, such as masking or transparency information.

Example

The following example returns whether the first shape on the first page of the active publication contains an alpha channel. If the picture is linked, and the original picture contains an alpha channel, that is also returned. This example assumes the shape is a picture.

```
With ActiveDocument.Pages(1).Shapes(1).PictureFormat
    If .HasAlphaChannel = msoTrue Then
        Debug.Print .Filename
        Debug.Print "This picture contains an alpha channel."

        If .IsLinked = msoTrue Then
            If .OriginalHasAlphaChannel = msoTrue Then
                Debug.Print "The linked picture " & _
                "also contains an alpha channel."
            End If
        End If
    End If
End With
```




Show All

HasNextLink Property

MsoTrue if the text frame has a valid forward text box link. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse Indicates the specified text box does not have a forward text box link.

msoTriStateMixed

msoTriStateToggle

msoTrue Indicates the specified text box has a forward text box link.

expression.**HasNextLink**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example breaks all links in the document to the first specified text frame if links exist. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub AddPreviousNextLinkPages()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame
        If .HasNextLink Then .BreakForwardLink
        If .HasPreviousLink Then .PreviousLinkedTextFrame _
            .BreakForwardLink
    End With
End Sub
```


Show All

HasPreviousLink Property

MsoTrue if the text frame has a valid link to a backward text box. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse Indicates the specified text box does not have a backward text box link.

msoTriStateMixed

msoTriStateToggle

msoTrue Indicates the specified text box has a backward text box link.

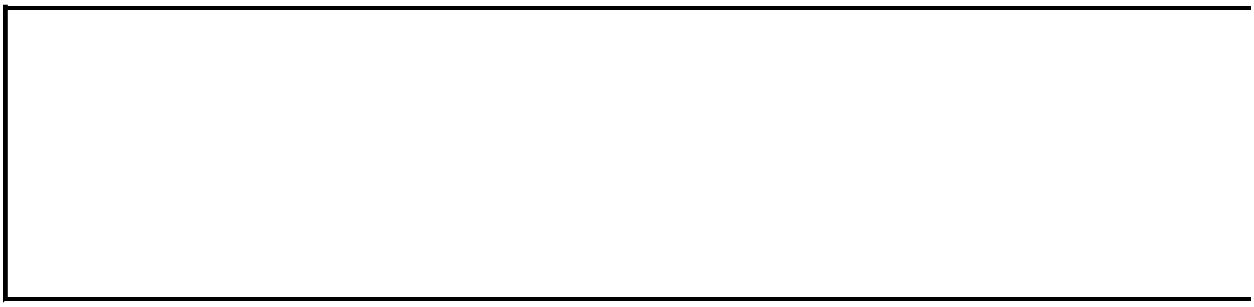
expression.**HasPreviousLink**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example breaks all links in the document to the first specified text frame if links exist. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub AddPreviousNextLinkPages()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame
        If .HasNextLink Then .BreakForwardLink
        If .HasPreviousLink Then .PreviousLinkedTextFrame _
            .BreakForwardLink
    End With
End Sub
```




Show All

HasTable Property

Returns **msoTrue** if the shape represents a **TableFrame** object or **msoFalse** if the shape represents any other object type. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shapes in the range do not represent a **TableFrame** object.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The shapes in the range represent a **TableFrame** object.

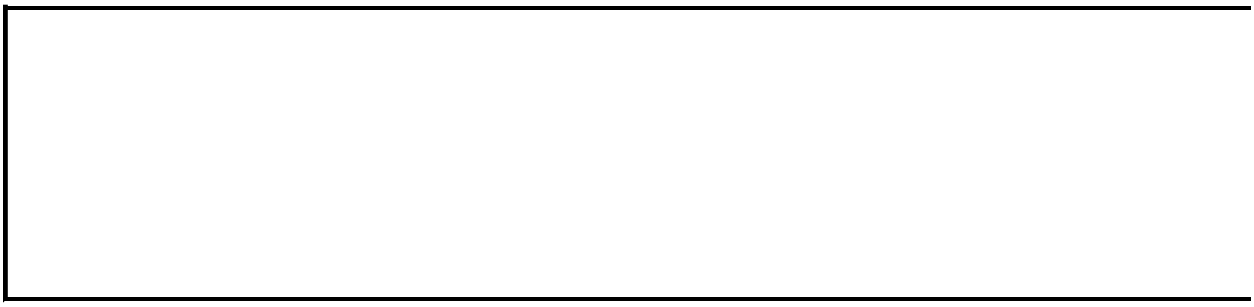
expression.**HasTable**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example checks the currently selected shape to see if it is a table. If it is, the code sets the width of column one to one inch (72 points).

```
Sub IsTable()

    With Application.Selection.ShapeRange
        If .HasTable = msoTrue Then
            .Table.Columns(1).Width = 72
        End If
    End With

End Sub
```




Show All

HasText Property

 HasText property as it applies to the **Cell** object.

Returns a **Boolean** value indicating whether the specified cell contains any text. **True** if the specified cell contains text. Read-only.

expression.**HasText**

expression Required. An expression that returns a **Cell** object.

 HasText property as it applies to the **TextFrame** object.

Returns an **MsoTriState** constant indicating whether the specified shape has text associated with it. Read-only.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this property.
msoFalse The specified shape does not have text associated with it.
msoTriStateMixed Not used with this property.
msoTriStateToggle Not used with this property.
msoTrue The specified shape has text associated with it.

expression.**HasText**

expression Required. An expression that returns a **TextFrame** object.

Example

 As it applies to the **Cell** object.

If shape one on page one contains a table and the first cell of the table contains text, this example displays the text in a message box.

```
With ActiveDocument.Pages(1).Shapes(1)

    ' Check for table.
    If .HasTable Then
        With .Table.Cells(StartRow:=1, StartColumn:=1, _
                EndRow:=1, EndColumn:=1).Item(1)

            ' Check for text in first cell.
            If .HasText Then
                MsgBox "Text from first cell of table: " _
                    & vbCr & .Text
            Else
                MsgBox "No text in first cell."
            End If

        End With
    Else
        MsgBox "No table in shape one."
    End If

End With
```

 As it applies to the **TextFrame** object.

If shape two on the first page of the active publication contains text, this example resizes the shape to fit the text.

```
With ActiveDocument.Pages(1).Shapes(2).TextFrame
    If .HasText Then .AutoFitText = pbTextAutoFitBestFit
End With
```


Show All

HasTextFrame Property

Returns an **MsoTriState** constant if the specified shape has a **TextFrame** object associated with it. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The specified shape does not have a **TextFrame** object associated with it.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified shape has a **TextFrame** object associated with it.

expression.**HasTextFrame**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the **HasTextFrame** property is true, clients must check the value of the **HasText** property of the **TextFrame** object to determine if there is any text on the shape.

Example

This example tests all the shapes in the selection and if none have text frames associated with them, they are left aligned.

```
Sub MoveLeft()

    Dim shpAll As ShapeRange

    Set shpAll = Application.ActiveDocument.Selection.ShapeRange
    If shpAll.HasTextFrame = msoFalse Then
        shpAll.Align msoAlignLefts, msoTrue
    End If

End Sub
```

HasTransparencyColor Property

Returns a **Boolean** that indicates whether a transparency color has been applied to the specified picture. Read-only.

expression.**HasTransparencyColor**()

expression Required. An expression that returns a **PictureFormat** object.

Example

The following example returns a list of the pictures with transparency colors in the active publication.

```
Sub ListPicturesWithTransColors()
Dim pgLoop As Page
Dim shpLoop As Shape

    For Each pgLoop In ActiveDocument.Pages
        For Each shpLoop In pgLoop.Shapes

            If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPi

                With shpLoop.PictureFormat
                    If .IsEmpty = msoFalse Then
                        If .HasTransparencyColor = True Then
                            Debug.Print .Filename
                        End If
                    End If
                End With

            End If

        Next shpLoop
    Next pgLoop

End Sub
```

Header Property

Returns a **HeaderFooter** object representing the header of the specified **Page** object. Read only.

expression.**Header**

expression Required. An expression that returns a **Page** object from the **MasterPages** collection.

Remarks

This property is for master pages only. A "This feature is only for master pages" error is returned if the header property is accessed from a **Page** object that is returned form the **Pages** collection. A new **HeaderFooter** object is created for the specified master page by accessing this property.

Example

The following example creates a **HeaderFooter** object and sets it to the header of the first master page.

```
Dim objHeader As HeaderFooter
Set objHeader = ActiveDocument.MasterPages(1).Header
```

The **HeaderFooter** object returned by the **Header** Property can be used to manipulate the header content. The following example sets some properties of the **HeaderFooter** object of the first master page,

```
With ActiveDocument.masterPages(1)
    With .Header
        .TextRange.Text = "Windows" & Chr(13) & "Office" & Chr(13) &
        With .TextRange.ParagraphFormat
            .SetListType Value:=pbListTypeBullet, BulletText:="*"
            .Alignment = pbParagraphAlignmentLeft
        End With
    End With
    With .Footer
        .TextRange.Hyperlinks.Add Text:=.TextRange, _
            Address:="http://www.tailspintoys.com", _
            TextToDisplay:="Tailspin"
    End With
End With
```


Show All

Height Property

 **Height** property as it applies to the **ReaderSpread** and **PrintableRect** objects.

Returns a **Single** that represents the height, in points, of the page (for the **ReaderSpread** object) or the printable rectangle (for the **PrintablePect** object). Read-only.

expression.**Height**

expression Required. An expression that returns one of the above objects.

 **Height** property as it applies to the **Label** object.

Returns a **Variant** that represents the height (in points) of the label. Read-only.

expression.**Height**

expression Required. An expression that returns one of the above objects.

 **Height** property as it applies to the **Window** object.

Returns or sets a **Long** that represents the height (in points) of the window. Read/write.

expression.**Height**

expression Required. An expression that returns one of the above objects.

 **Height** property as it applies to the **Cell**, **CellRange**, and **Page** objects.

Returns a **Long** that represent the height (in points) of a cell, range of cells, or page. Read-only.

expression.**Height**

expression Required. An expression that returns one of the above objects.

Height property as it applies to the **Row** and **Shape** objects.

Returns or sets a **Variant** that represents the height (in points) of a specified table row or shape. Read/write.

expression.**Height**

expression Required. An expression that returns one of the above objects.

Height property as it applies to the **ShapeRange** object.

Returns a **Variant** that represents the height (in points) of a specified range of shapes. Read-only.

expression.**Height**

expression Required. An expression that returns one of the above objects.

Height property as it applies to the **PictureFormat** object.

Returns a **Variant** that represents the height, in points, of the specified picture or OLE object. Read-only.

expression.**Height**

expression Required. An expression that returns a **PictureFormat** object.

Remarks

The valid range for the **Height** property depends on the size of the application workspace and the position of the object within the workspace. For centered objects on non-banner page sizes, the **Height** property may be 0.0 to 50.0 inches. For centered objects on banner page sizes, the **Height** property may be 0.0 to 241.0 inches.

Example

 As it applies to the **Window** object.

This example sets the height and width of the active window if the window is neither maximized nor minimized.

```
Sub SetWindowHeight()
    With ActiveWindow
        If .WindowState <> pbWindowStateNormal Then
            .WindowState = pbWindowStateNormal
            .Height = InchesToPoints(5)
            .Width = InchesToPoints(5)
        End If
    End With
End Sub
```

 As it applies to the **Row** object.

This example creates a new table and sets the height and width of the second row and column, respectively.

```
Sub SetRowHeightColumnWidth()
    With ActiveDocument.Pages(1).Shapes.AddTable(NumRows:=3, _
            NumColumns:=3, Left:=80, Top:=80, Width:=400, Height:=12
            .Rows(2).Height = 72
            .Columns(2).Width = 72
    End With
End Sub
```

HiddenFields Property

Returns a **WebHiddenFields** object that represents hidden Web fields attached to a Submit command button.

expression.**HiddenFields**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a new hidden Web field to a new Submit command button.

```
Sub CreateActionWebButton()
    With ActiveDocument.Pages(1).Shapes
        With .AddWebControl _
                (Type:=pbWebControlCommandButton, Left:=150, _
                Top:=150, Width:=75, Height:=36).WebCommandButton
            .ButtonText = "Submit"
            .ButtonType = pbCommandButtonSubmit
        End With
        .Item(1).WebCommandButton.HiddenFields.Add _
            Name:="User", Value:="PowerUser"
    End With
End Sub
```


Show All

HorizontalAlignment Property

Sets or returns a **PbWizardNavBarAlignment** constant that represents the horizontal alignment of the buttons in a Web navigation bar set. Read/write.

HorizontalAlignment property can be set to any of these **PbWizardNavBarAlignment** constants:

pbnbAlignCenter

pbnbAlignLeft

pbnbAlignRight

expression.**HorizontalAlignment**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Remarks

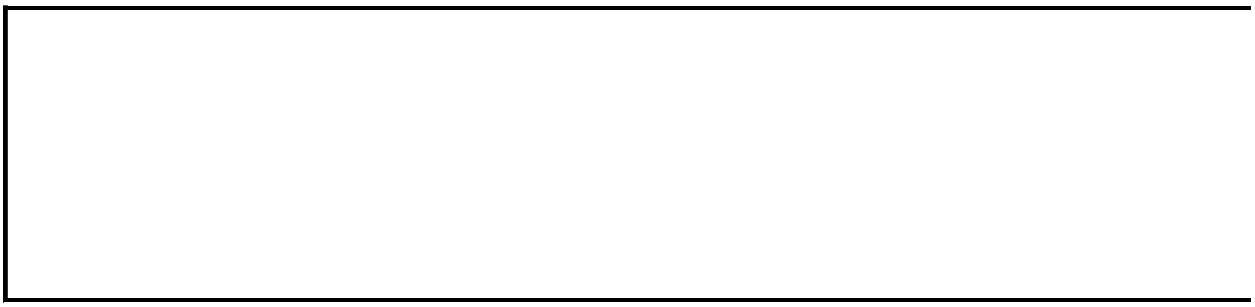
This property is used to set the way that buttons are displayed in a horizontally oriented Web navigation bar set. For example, a **WebNavigationBarSet** containing 5 links with the **HorizontalButtonCount** property set to **3** and the **HorizontalAlignment property** set to **right** will align the buttons in a grid of 3 columns and 1 row. The first 3 buttons will be in the first row and the remaining 2 buttons will be in the rightmost columns of the second row.

Returns "Access denied" if **IsHorizontal** = **False** for the specified **WebNavigationBarSet** object. Use the **ChangeOrientation** method to set the orientation of the Web navigation bar set to horizontal first before setting the **HorizontalAlignment** property.

Example

The following example returns the first Web navigation bar set from the active document, changes the orientation to **horizontal** if necessary, sets the **HorizontalButtonCount** property to **3**, and then sets the **HorizontalAlignment** property to **pbnbAlignRight**.

```
With ActiveDocument.WebNavigationBarSets(1)
    If .IsHorizontal = False Then
        .ChangeOrientation pbNavBarOrientHorizontal
    End If
    .HorizontalButtonCount = 3
    .HorizontalAlignment = pbnbAlignRight
End With
```



HorizontalBaseLineOffset Property

Returns a **Single** that represents the horizontal baseline offset of the specified **LayoutGuides** object. Read/Write.

expression.**HorizontalBaseLineOffset**

expression Required. An expression that returns a **LayoutGuides** object.

Remarks

When setting the layout guide properties of a **Page** object it must be returned from the **MasterPages** collection.

Example

This example sets the horizontal baseline offset of the layout guides object to 12 for the second master page in the active document.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.MasterPages(2).LayoutGuides
objLayout.HorizontalBaseLineSpacing = 12
```

Setting the layout guide properties for the active document will only affect the first master page. This example sets the horizontal baseline offset of the active document's layout guides to 12, affecting only the first master page .

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.LayoutGuides
objLayout.HorizontalBaseLineOffset = 12
```

HorizontalBaseLineSpacing Property

Returns a **Single** that represents the horizontal baseline spacing of the specified **LayoutGuides** object. Read/write.

expression.**HorizontalBaseLineSpacing**

expression Required. An expression that returns a **LayoutGuides** object.

Remarks

When setting the layout guide properties of a **Page** object it must be returned from the **MasterPages** collection.

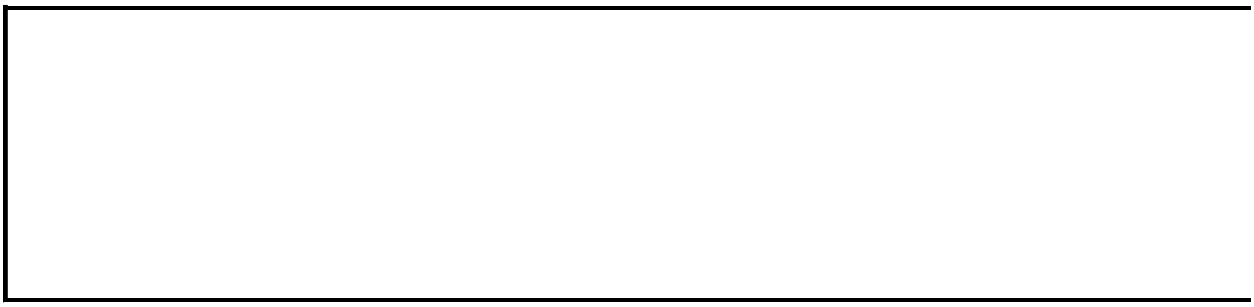
Example

This example sets the horizontal baseline spacing of the layout guides object to 20 for the second master page in the active document.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.MasterPages(2).LayoutGuides
objLayout.HorizontalBaseLineSpacing = 20
```

Setting the layout guide properties for the active document will only affect the first master page. This example sets the horizontal baseline spacing of the active document's layout guides to 20, affecting only the first master page .

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.LayoutGuides
objLayout.HorizontalBaseLineSpacing = 20
```



HorizontalButtonCount Property

Sets or returns a **Long** representing the number of buttons in each row of buttons for a Web navigation bar set. Read/write. **Long**.

expression.**HorizontalButtonCount**

expression Required. An expression that returns a **WebNavigationBarSet** object.

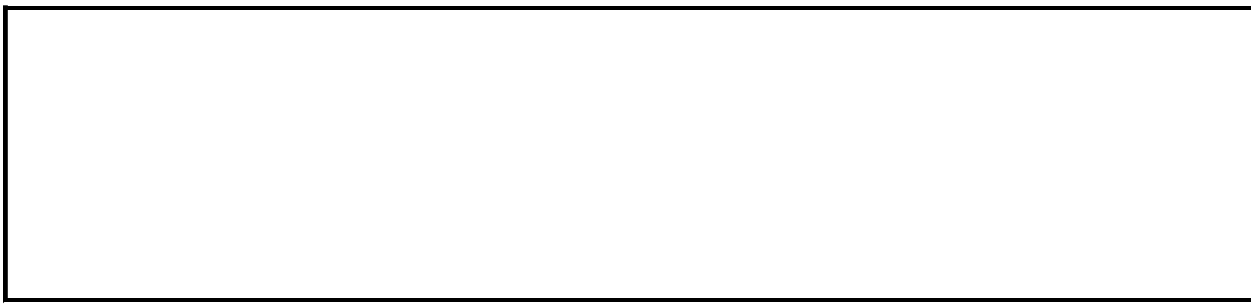
Remarks

Returns "Access denied" if **IsHorizontal** = **False** for the specified **WebNavigationBarSet** object. Use the **ChangeOrientation** method to set the orientation of the Web navigation bar set to **horizontal** first before setting the **HorizontalButtonCount** property.

Example

The following example returns the first Web navigation bar set from the active document, changes the orientation to **horizontal** if necessary, sets the **HorizontalButtonCount** property to **3**, and then sets the **HorizontalAlignment** property to **pbnbAlignLeft**.

```
With ActiveDocument.WebNavigationBarSets(1)
    If .IsHorizontal = False Then
        .ChangeOrientation pbNavBarOrientHorizontal
    End If
    .HorizontalButtonCount = 3
    .HorizontalAlignment = pbnbAlignRight
End With
```




Show All

HorizontalFlip Property

 **HorizontalFlip** property as it applies to the **Shape** and **ShapeRange** objects.

Returns **msoTrue** if the specified shape has been flipped around its horizontal axis. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shape has not been flipped around its horizontal axis.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The shape has been flipped around its horizontal axis.

expression.**HorizontalFlip**

expression Required. An expression that returns one of the objects in the Applies To list.

 **HorizontalFlip** property as it applies to the **AdvancedPrintOptions** object.

True to print a horizontally mirrored image of the specified publication. The default is **False**. Read/write **boolean**.

expression.**HorizontalFlip**

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property is only accessible if the active printer is a PostScript printer. Returns a run-time error if a non-PostScript printer is specified. Use the **IsPostscriptPrinter** property of the **AdvancedPrintOptions** object to determine if the specified printer is a PostScript printer.

This property is saved as an application setting and applied to future instances of Publisher.

This property corresponds to the **Flip horizontally** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

This property is mostly used when printing to film on an imagesetter so that the image reads correctly when the emulsion side of the film is down (as when burning a press plate).

Example

 As it applies to the **Shape** and **ShapeRange** objects.

This example restores each shape on the active publication to its original state if it has been flipped horizontally or vertically.

```
Sub Flipper()

    Dim shpS As Shape

    For Each shpS In ActiveDocument.MasterPages.Item(1).Shapes
        If shpS.HorizontalFlip = msoTrue Then shpS.Flip msoFlipHoriz
        If shpS.VerticalFlip = msoTrue Then shpS.Flip msoFlipVertica
    Next

End Sub
```

 As it applies to the **AdvancedPrintOptions** object.

The following example determines if the active printer is a PostScript printer. If it is, the active publication is set to print as a horizontally and vertically mirrored, negative image of itself.

```
Sub PrepToPrintToFilmOnImagesetter()

With ActiveDocument.AdvancedPrintOptions
    If .IsPostscriptPrinter = True Then
        .HorizontalFlip = True
        .VerticalFlip = True
        .NegativeImage = True
    End If
End With

End Sub
```

HorizontalGap Property

Returns or sets a **Variant** indicating the distance between the right edge of the publication page and left edge of the next publication page in the same row. Numeric values are evaluated as points; string values may be in any unit supported by Publisher (for example, "2.5 in"). Valid range is zero to the difference between the sheet width and the page width. Read/write.

expression.**HorizontalGap**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

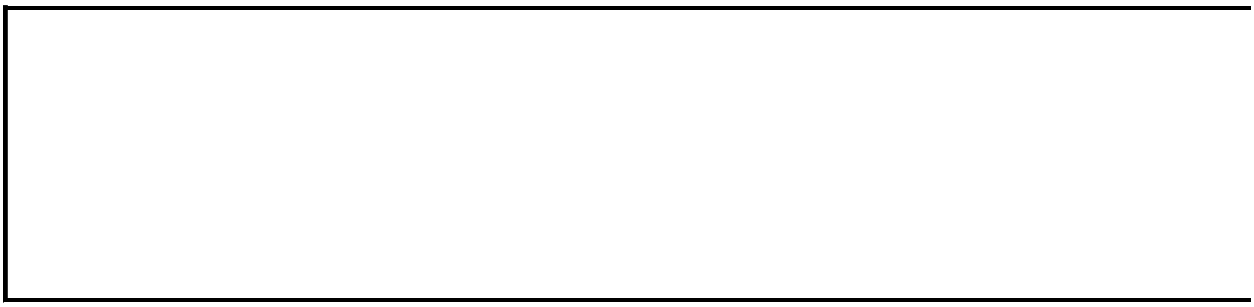
This property applies only to publications where multiple pages will be printed on each printer sheet. Using this property for any other publication will generate an error.

When used with the **Label** object, the **HorizontalGap** property is read/write only when accessed from **.PageSetup.Label**. Otherwise, it is read-only.

Example

The following example sets the horizontal distance between publication pages that will be printed on the same sheet to 96 points.

```
Sub SetHorizontalGap()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(8)
        .PageWidth = InchesToPoints(4)
        .MultiplePagesPerSheet = True
        .HorizontalGap = InchesToPoints(0.5)
    End With
End Sub
```




Show All

HorizontalPictureLocking Property

Returns or sets a **PbHorizontalPictureLocking** constant indicating where newly inserted pictures appear in relation to the specified frame. Read/write.

PbHorizontalPictureLocking can be one of these PbHorizontalPictureLocking constants.

pbHorizontalLockingLeft New pictures are inserted along the left edge of the frame.

pbHorizontalLockingNone New pictures are inserted in the middle between the left and right edges of the frame.

pbHorizontalLockingRight New pictures are inserted along the right edge of the frame.

pbHorizontalLockingStretch New pictures are horizontally stretched to the full width of the frame.

expression.**HorizontalPictureLocking**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example locks the specified picture to the top left corner of the picture frame. Shape one on page one of the active publication must be a picture frame for this example to work.

```
With ActiveDocument.Pages(1).Shapes(1).PictureFormat
    .HorizontalPictureLocking = pbHorizontalLockingLeft
    .VerticalPictureLocking = pbVerticalLockingTop
End With
```


Show All

HorizontalRepeat Property

Returns a **Long** that represents the number of times the catalog merge area will repeat across the target publication page when the catalog merge is executed. Read-only.

expression.**HorizontalRepeat**

expression Required. An expression that returns a **CatalogMergeShapes** object.

Remarks

When the catalog merge is executed, the catalog merge area repeats once for each selected record in the specified data source.

The number of times the catalog merge area repeats across the page is determined by the width of the area. Use the **Width** property of the **Shape** object to return or set the horizontal size of the catalog merge area.

The **VerticalRepeat** property of the **CatalogMergeShapes** object represents the number of times the catalog merge area repeats vertically down the target publication page.

Example

The following example returns the number of times the catalog merge area will repeat horizontally and vertically on the target publication page when the catalog merge is performed. This example assumes the catalog merge area is the first shape on the first page of the specified publication.

```
Sub CatalogMergeDimensions()

    With ThisDocument.Pages(1).Shapes(1)
        Debug.Print .Width
        Debug.Print .CatalogMergeItems.HorizontalRepeat
        Debug.Print .Height
        Debug.Print .CatalogMergeItems.VerticalRepeat
    End With

End Sub
```

HorizontalScale Property

Returns a **Long** that represents the scaling of the picture along its horizontal axis. The scaling is expressed as a percentage (for example, 200 equals 200% scaling). Read-only.

expression.**HorizontalScale**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

The effective resolution of a picture is inversely proportional to the scaling at which the picture is printed. The larger the scaling, the lower the effective resolution. For example, suppose a picture measuring 4 inches by 4 inches was originally scanned at 300 dpi. If that picture is scaled to 2 inches by 2 inches, its effective resolution is 600 dpi.

Use the **EffectiveResolution** property of the **PictureFormat** object to determine the resolution at which the picture or OLE object will print in the specified document.

Example

The following example prints selected image properties for each picture in the active publication.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPictur

            With shpLoop.PictureFormat

                  If .IsEmpty = msoFalse Then
                       Debug.Print "File Name: " & .Filename
                       Debug.Print "Resolution in Publication: " &
                       Debug.Print "Horizontal Scaling: " & .Horizo
                       Debug.Print "Height in publication: " & .Hei
                       Debug.Print "Vertical Scaling: " & .Vertical
                       Debug.Print "Width in publication: " & .Widt
                  End If

            End With
        End If
    Next shpLoop
Next pgLoop
```

Hwnd Property

Returns a **Long** indicating the handle to the Publisher application window. Read-only.

expression.**Hwnd**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the handle to the Publisher application window.

```
MsgBox "The handle to the Publisher application window is " & _
    Application.ActiveWindow.Hwnd
```

Hyperlink Property

Returns a **Hyperlink** object representing the hyperlink associated with the specified shape.

expression.**Hyperlink**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets shape one on page one in the active publication to jump to the specified Web site when the shape is clicked.

```
Dim hypTemp As Hyperlink

Set hypTemp = ActiveDocument.Pages(1).Shapes(1).Hyperlink

hypTemp.Address = "http://www.tailspintoys.com/"
```

Hyperlinks Property

Returns a **Hyperlinks** collection representing all the hyperlinks in the specified text range.

expression.**Hyperlinks**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example looks for all the shapes on page one of the active publication that have text frames and reports how many hyperlinks each shape has.

```
Dim hypAll As Hyperlinks
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    If shpLoop.HasTextFrame = msoTrue Then
        Set hypAll = shpLoop.TextFrame.TextRange.Hyperlinks
        Debug.Print "Shape " & shpLoop.Name _
            & " has " & hypAll.Count & " hyperlinks."
    End If
Next shpLoop
```

HyphenationZone Property

Returns or sets a **Variant** that represents the maximum amount of space that Microsoft Publisher leaves between the end of the last word in a line and the right margin. Read/write.

expression.**HyphenationZone**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example turns on automatic hyphenation and specifies the maximum amount of space between the end of the last word and the right margin equal to one inch (72 points).

```
Sub SetHyphenationZone()
    With Options
        .AutoHyphenate = True
        .HyphenationZone = 72
    End With
End Sub
```

ID Property

Returns a **Long** that represents the type of a shape, range of shapes, or property, type, or value of a wizard. Read-only.

expression.**ID**

expression Required. An expression that returns one of the above objects.

Example

This example displays the type for each shape on the first page of the active publication.

```
Sub ShapeID()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        MsgBox shp.ID
    Next shp
End Sub
```

IgnoreMaster Property

True for Publisher to ignore the master page formatting for the specified page. Read/write **Boolean**.

expression.**IgnoreMaster**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a red star in the upper left corner of the master page so that it shows up on each page; then it adds a couple of new pages and sets one of the pages to ignore the master page so that the shape doesn't show on it.

```
Sub AddNewPageIgnoreMaster()
    Dim pgNew As Page

    With ActiveDocument
        .MasterPages(1).Shapes.AddShape(Type:=msoShape5pointStar, _
            Left:=50, Top:=50, Width:=50, Height:=50).Fill.ForeColor
            .CMYK.SetCMYK Cyan:=0, Magenta:=255, Yellow:=255, Black:
        .Pages.Add Count:=1, After:=1
        Set pgNew = .Pages.Add(Count:=1, After:=1)
        pgNew.IgnoreMaster = True
    End With
End Sub
```


Show All

ImageFormat Property

Returns a **PbImageFormat** constant that represents the image format of a picture as determined by Microsoft® Windows® Graphics Device Interface (GDI+). Read-only.

PbImageFormat can be one of these PbImageFormat constants.
pbImageFormatCMYKJPEG (See Remarks.)
pbImageFormatDIB (See Remarks.)
pbImageFormatEMF (See Remarks.)
pbImageFormatGIF (See Remarks.)
pbImageFormatJPEG
pbImageFormatPICT (See Remarks.)
pbImageFormatPNG
pbImageFormatTIFF
pbImageFormatUNKNOWN
pbImageFormatWMF

expression.**ImageFormat**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

The **ImageFormat** property applies to the original picture, rather than the placeholder picture, if there is one.

The **ImageFormat** property indicates the format of the picture after it has been imported into the Windows environment, rather than its original file format. If the picture's file format is not natively supported by the Windows operating system, the picture is converted to an analogous format that is natively supported. As a result, the **pbImageFormatCMYKJPEG**, **pbImageFormatDIB**, **pbImageFormatEMF**, **pbImageFormatGIF**, and **pbImageFormatPICT** constants will rarely, if ever, be returned. Consult the table below for specific file format conversions.

File format	Constant returned
.bmp, .dib, .gif, .pict	pbImageFormatPNG
.emf, .eps, .epfs	pbImageFormatWMF
CMYK .jfif, .jpeg, .jpg	pbImageFormatJPEG

Windows GDI+ is the portion of the Windows XP operating system and the Windows Server 2003 operating system that provides two-dimensional vector graphics, imaging, and typography.

Example

The following example prints a list of the .jpg and .jpeg images present in the active publication.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes

        If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPictur

            With shpLoop.PictureFormat
                If .IsEmpty = msoFalse Then

                    If .ImageFormat = pbImageFormatJPEG Then
                        Debug.Print .Filename
                    End If

                End If
            End With

        End If

    Next shpLoop
Next pgLoop
```

Included Property

True if a record is included in a mail merge. Read/write **Boolean**.

expression.**Included**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **SetAllIncludedFlags** method to set the included status for all mail merge records.

Example

This example searches the records to verify that the length of the PostalCode field for each record is at least five digits long. If it is not, the record is excluded from the mail merge and flagged as invalid.

```
Sub ExcludeRecords()
    Dim intRecord As Integer
    With ActiveDocument.MailMerge
        For intRecord = 1 To .DataSource.RecordCount
            .DataSource.ActiveRecord = intRecord
            If Len(.DataSource.DataFields("PostalCode").Value) < 5 T
                With .DataSource
                    .Included = False
                    .InvalidAddress = True
                    .InvalidComments = "This record is removed " & _
                        "from the mail merge because its postal code
                        "has less than five digits."
                End With
            End If
        Next
    End With
End Sub
```

IncludePageOnNewWebNavigationBa Property

Returns or sets a **Boolean** value that specifies whether a link to a Web page will be added to the automatic navigation bars of new pages. Read/write.

expression.**IncludePageOnNewWebNavigationBars**

expression Required. An expression that returns a **WebPageOptions** object.

Remarks

The default value of the **IncludePageOnNewWebNavigationBars** property is **False**, which means that links to the specified page will not be added to the automatic navigation bars of new pages.

Setting this property to **False** does not remove links to the specified page from any automatic navigation bars that already include them, but it does prevent links to the page from being added to automatic navigation bars of new pages.

Setting this property to **True** applies only to automatic navigation bars of new pages, and does not update existing automatic navigation bars within the Web publication.

When adding a new page to the Web publication by using the **Pages.Add** method, the optional ***AddHyperlinkToWebNavBar*** parameter can be used to specify whether links to the new page will be added to existing automatic navigation bars. The value of this parameter is used to populate the value of the **IncludePageOnNewWebNavigationBars** property.

Example

The following example specifies that links to page two of the active Web publication should be added to the automatic navigation bars of new pages. Note that if a new page is added to the publication after this point, the **IncludePageOnNewWebNavigationBars** property will be **False**.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(2).WebPageOptions
With theWPO
    .IncludePageOnNewWebNavigationBars = True
End With
```

The following example demonstrates adding two new pages to the publication by using the **Pages.Add** method. The ***AddHyperlinkToWebNavBar*** parameter is set to **True**, which specifies that links to these two new pages be added to the automatic navigation bars of existing pages.

Another page is then added to the publication, and the ***AddHyperlinkToWebNavBar*** is omitted. This means that the **IncludePageOnNewWebNavigationBars** property is **False** for the newly added page, and links to this page will not be included in the automatic navigation bars of existing pages.

```
Dim thePage As page
Dim thePage2 As page

Set thePage = ActiveDocument.Pages.Add(Count:=2, _
    After:=4, AddHyperlinkToWebNavBar:=True)

Set thePage2 = ActiveDocument.Pages.Add(Count:=1, After:=6)
```

Index Property

Returns a **Long** that represents the position of a particular item in a specified collection. Read-only.

expression.**Index**

expression Required. An expression that returns one of the objects in the Applies To list.

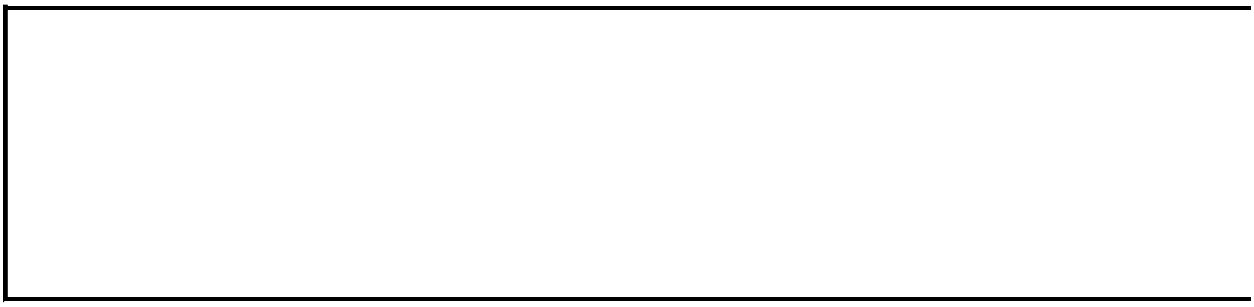
Example

The following example loops through the **MailMergeDataFields** collection and displays the **Index** and **Name** properties for each field.

```
Dim mmfLoop As MailMergeDataField

With ActiveDocument.MailMerge.DataSource
    If .DataFields.Count > 0 Then
        For Each mmfLoop In .DataFields
            Debug.Print "Field " & mmfLoop.Name _
                & " / Index " & mmfLoop.Index
        Next mmfLoop
    Else
        Debug.Print "No fields to report."
    End If
End With
```

The following example loops through the **Plates** collection and displays the **Index** and **Name** properties for each plate.

```
Dim plaLoop As Plate

If ActiveDocument.Plates.Count > 0 Then
    For Each plaLoop In ActiveDocument.Plates
        Debug.Print "Plate " & plaLoop.Name _
            & " / Index " & plaLoop.Index
    Next plaLoop
Else
    Debug.Print "No plates to report."
End If
```



Ink Property

Returns or sets a **Long** indicating whether the specified color is a spot color, and if so, the spot plate to which it belongs. Valid values are **pbInkNone** (default; meaning that the color is not a spot color) or a number between 1 and *n* where *n* is the number of spot plates. Read/write.

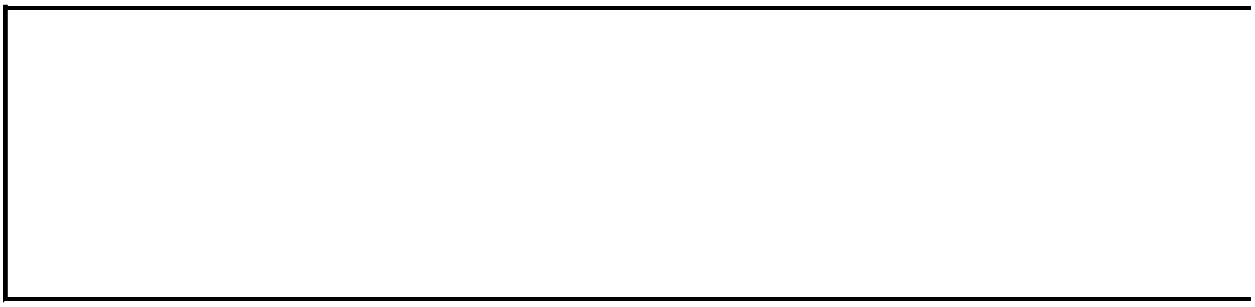
expression.**Ink**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example specifies that the color of the first text range on page one of the active publication should be assigned to spot plate two.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange.Font.Color.Ink = 2
```




Show All

InkName Property

Returns a **PbInkName** constant that represents the name of the ink to be printed using this plate. Read-only.

expression.**InkName**

expression Required. An expression that returns one of the objects in the Applies To list.

PbInkName can be one of these **pbInkName** constants.
pbInkNameBlack
pbInkNameCyan
pbInkNameMagenta
pbInkNameYellow
pbInkNameSpotColor1
pbInkNameSpotColor2
pbInkNameSpotColor3
pbInkNameSpotColor4
pbInkNameSpotColor5
pbInkNameSpotColor6
pbInkNameSpotColor7
pbInkNameSpotColor8
pbInkNameSpotColor9
pbInkNameSpotColor10
pbInkNameSpotColor11
pbInkNameSpotColor12

Remarks

Use the **FindPlateByInkName** method of the **PrintablePlates** collection to return a specific plate by referencing its ink name.

Example

The following example returns a list of the printable plates currently in the collection for the active publication. The example assumes that separations have been specified as the active publication's print mode.

```
Sub ListPrintablePlates()
    Dim pplTemp As PrintablePlates
    Dim pplLoop As PrintablePlate


    Set pplTemp = ActiveDocument.AdvancedPrintOptions.PrintablePlate
    Debug.Print "There are " & pplTemp.Count & " printable plates in

    For Each pplLoop In pplTemp
        With pplLoop
            Debug.Print "Printable Plate Name: " & .Name
            Debug.Print "Index: " & .Index
            Debug.Print "Ink Name: " & .InkName
            Debug.Print "Plate Angle: " & .Angle
            Debug.Print "Plate Frequency: " & .Frequency
            Debug.Print "Print Plate?: " & .PrintPlate
        End With
    Next pplLoop
End Sub
```


Show All

InksToPrint Property

Returns or sets a **PbInksToPrint** constant that represents which inks to print as separate plates. Read/write.

PbInksToPrint can be one of these PbInksToPrint constants.

pbInksToPrintAll *Default*. Print a separate plate for every ink defined for the publication, whether or not it is used.

pbInksToPrintConvertSpotToProcess Convert any spot color used in the publication to their equivalent CMYK values and print these objects as part of the process color separations.

pbInksToPrintused Print separate plates for only those inks used in the publication.

expression.**InksToPrint**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property is only accessible if separations are being printed. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specifiy that separations are to be printed. Returns "Permission Denied" if any other print mode is specified.

The **InksToPrint** property is equivalent to the **These Plates** control on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

The following example tests to determine if the active publication has been set to print as separations. If it has, it is set to print only plates for the inks actually used in the publication, and to not print plates for any pages where a color is not used.

```
Sub PrintOnlyInksUsed
        With ActiveDocument.AdvancedPrintOptions
                If .PrintMode = pbPrintModeSeparations Then
                        .InksToPrint = pbInksToPrintUsed
                        .PrintBlankPlates = False
                End If
        End With
End Sub
```


Show All

InlineAlignment Property

Returns or sets a **PbInlineAlignment** constant that indicates whether an inline shape has left, right, or in-text alignment. Read/write.

PbInlineAlignment can be one of these **PbInlineAlignment** constants.
pbInlineAlignmentCharacter
pbInlineAlignmentLeft
pbInlineAlignmentMixed
pbInlineAlignmentRight

expression.**InlineAlignment**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

An automation error is returned if the shape is not already inline.

Example

The following example moves the second shape on the second page of the publication into the text flow by using the **MoveIntoTextFlow** method. The **InlineAlignment** property is then used to align the shape to the right.

```
Dim theShape As Shape
Dim theRange As TextRange

Set theRange = ActiveDocument.Pages(2).Shapes(1).TextFrame.TextRange
Set theShape = ActiveDocument.Pages(2).Shapes(2)

If Not theShape.IsInline = msoTrue Then
    theShape.MoveIntoTextFlow Range:=theRange
    theShape.InlineAlignment = pbInlineAlignmentRight
End If
```

InlineShapes Property

Returns an **InlineShapes** collection, which represents the inline shapes contained within a text range. Read-only.

expression.**InlineShapes**

expression Required. An expression that returns a **TextRange** object.

Remarks

Using **TextFrame.Story.TextRange.InlineShapes** will return all inline shapes in a text frame, including those that are in overflow. Using **TextFrame.TextRange.InlineShapes** will return only visible inline shapes in a text frame, and not those that are in overflow.

Example

The following example finds the first shape (a text box) on page one of the active publication. The **InlineShapes** property is then used to determine whether any inline shapes exist in the text box. If any are found, each inline shape is flipped vertically, and its fore color is set to red.

Note that by using **TextFrame.Story.TextRange.InlineShapes**, any inline shapes that are in overflow will also be found.

```
Dim theShape As Shape
Dim i As Integer

Set theShape = ActiveDocument.Pages(1).Shapes(1)

With theShape.TextFrame.Story.TextRange
    If .InlineShapes.Count > 0 Then
        For i = 1 To .InlineShapes.Count
            .InlineShapes(i).Flip (msoFlipVertical)
            .InlineShapes(i).Fill.ForeColor.RGB = vbRed
        Next
    End If
End With
```

InlineTextRange Property

Returns a **TextRange** object that reflects the position of the inline shape in its containing text range. Read-only.

expression.**InlineTextRange**

expression Required. An expression that returns a **Shape** object. Note that the shape must be an inline shape contained within the **InlineShapes** collection.

Remarks

The returned text range will contain a single object representing the inline shape. An automation error is returned if the shape is not inline.

Example

The following example finds the first shape (a text box) on the first page of the publication, and determines if the text range within the text box contains inline shapes. If inline shapes are found, the **InlineTextRange** property is used to represent the inline shape after a block of text is inserted.

```
Dim theShape As Shape
Dim theTextRange As TextRange
Dim i As Integer

Set theShape = ActiveDocument.Pages(1).Shapes(1)

If Not theShape.IsInline = True Then
    With theShape.TextFrame.Story.TextRange
        If .InlineShapes.Count > 0 Then
            Set theTextRange = theShape.TextFrame.Story.TextRange
            For i = 1 To .InlineShapes.Count
                With .InlineShapes(i)
                    .InlineTextRange.InsertAfter (" (Figure " & i &
                End With
            Next
        End If
    End With
End If
```


Show All

InsetPen Property

Returns or sets an **MsoTriState** constant indicating whether a specified shape's lines are drawn inside its boundaries. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not supported.

msoFalse Lines are drawn directly on the specified shape's boundaries.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue *default* Lines are drawn inside the specified shape's boundaries.

expression.**InsetPen**

expression Required. An expression that returns a **LineFormat** object.

Remarks

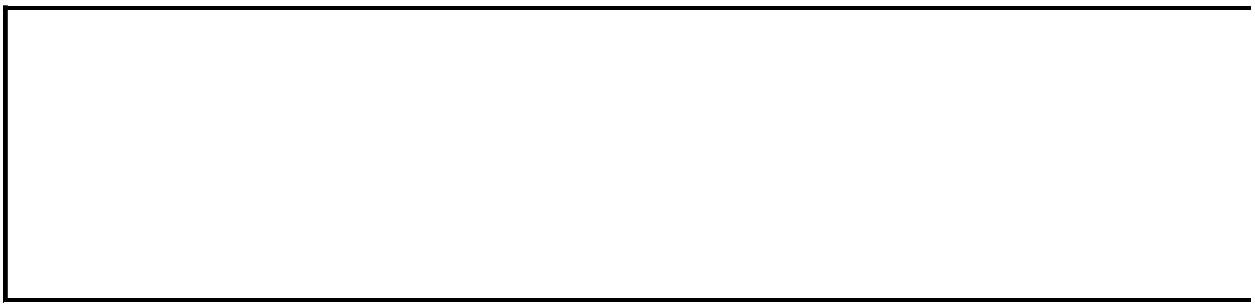
An error occurs if you attempt to set this property to **msoTrue** for any Office AutoShape which does not support inset pen drawing.

The value of the **InsetPen** property for tables is always **msoTrue**; attempting to set the property to any other value results in an error.

Example

The following example adds two rectangles to page one of the active publication, the first with its lines drawn inside its boundaries, and the second with its lines drawn on its boundaries.

```
Dim shpNew As Shape

With ActiveDocument.Pages(1).Shapes
    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=200, Top:=150, Width:=150, Height:=100)
    With shpNew.Line
        .Weight = 24
        .InsetPen = msoTrue
    End With

    Set shpNew = .AddShape(Type:=msoShapeRectangle, _
        Left:=200, Top:=300, Width:=150, Height:=100)
    With shpNew.Line
        .Weight = 24
        .InsetPen = msoFalse
    End With
End With
```



InUse Property

Returns **True** if the specified ink (represented by the plate) is used in the publication. Read-only **Boolean**.

expression.**InUse**

expression Required. An expression that returns a **Plate** object.

Remarks

This property corresponds to the **In Use** or **Not In Use** notation listed by each ink on the **Ink** tab of the **Color Printing** dialog box.

Example

The following example loops through the active publication's plates collection, determines which plates represent inks that are not used in the publication, and deletes them.

```
Sub DeleteUnusedInks()

Dim intCount As Integer

With ActiveDocument.Plates
    For intCount = .Count To 1 Step -1
        With .Item(intCount)
            If .InUse = False Then
                Debug.Print "Name: " & .Name
                .Delete
            End If
        End With
    Next
End With

End Sub
```

InvalidAddress Property

True to mark a record in a mail merge data source if it contains invalid data. Read/write **Boolean**.

expression.**InvalidAddress**

expression Required. An expression that returns one of the objects in the Applies To list.

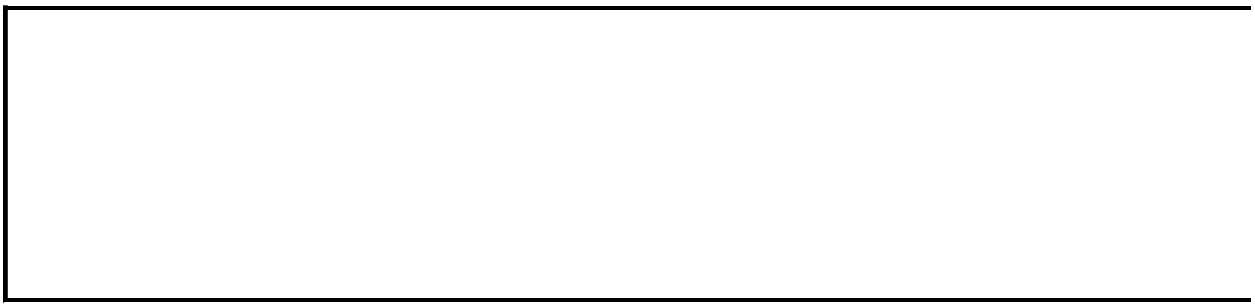
Remarks

Use the **SetAllErrorFlags** method to set both the **InvalidAddress** and **InvalidComments** properties for all records in a data source.

Example

This example searches the records to verify that the length of the PostalCode field for each record is at least five digits long. If it is not, the record is excluded from the mail merge and flagged as invalid.

```
Sub ExcludeRecords()
    Dim intRecord As Integer
    With ActiveDocument.MailMerge
        For intRecord = 1 To .DataSource.RecordCount
            .DataSource.ActiveRecord = intRecord
            If Len(.DataSource.DataFields("PostalCode").Value) < 5 T
                With .DataSource
                    .Included = False
                    .InvalidAddress = True
                    .InvalidComments = "This record is removed " & _
                        "from the mail merge because its postal code
                        "has less than five digits."
                End With
            End If
        Next
    End With
End Sub
```



InvalidComments Property

If the **InvalidAddress** property is **True**, this property returns or sets a **String** that describes invalid data in a mail merge record. Read/write.

expression.**InvalidComments**

expression Required. An expression that returns one of the objects in the Applies To list.

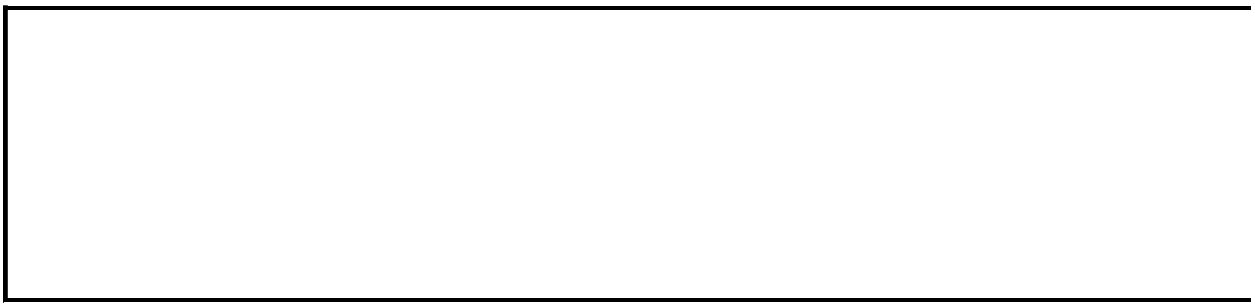
Remarks

Use the **SetAllErrorFlags** method to set both the **InvalidAddress** and **InvalidComments** properties for all records in a data source.

Example

This example searches the records to verify that the length of the PostalCode field for each record is at least five digits long. If it is not, the record is excluded from the mail merge and flagged as invalid.

```
Sub ExcludeRecords()
    Dim intRecord As Integer
    With ActiveDocument.MailMerge
        For intRecord = 1 To .DataSource.RecordCount
            .DataSource.ActiveRecord = intRecord
            If Len(.DataSource.DataFields("PostalCode").Value) < 5 T
                With .DataSource
                    .Included = False
                    .InvalidAddress = True
                    .InvalidComments = "This record is removed " & _
                        "from the mail merge because its postal code
                        "has less than five digits."
                End With
            End If
        Next
    End With
End Sub
```




Show All

IsDataSourceConnected Property

True if the specified publication is connected to a data source. Read-only.

expression.**IsDataSourceConnected**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

A publication must be connected to a valid data source to perform a mail merge or catalog merge.

Example

The following example tests whether the publication is connected to a data source and, if it is not, specifies and connects a data source to the publication. (Note that *PathToFile* must be replaced with a valid file path, and *TableName* with a valid data source table name, for this example to execute properly.)

```
Dim strDataSource As String
Dim strDataSourceTable As String

    'Specify data source and table name

    strDataSource = "PathToFile"
    strDataSourceTable = "TableName"

    'Connect to a datasource
    If Not (ThisDocument.IsDataSourceConnected) Then
        ThisDocument.MailMerge.OpenDataSource strDataSource, , strDa

    End If
```


Show All

IsEmpty Property

Returns a **MsoTriState** constant that represents whether the specified shape is an empty picture frame. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shape is not a empty picture frame.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified shape is an empty picture frame.

expression.**IsEmpty**()

expression Required. An expression that returns a **PictureFrame** object.

Example

The following example tests each picture in the active publication, and if it is not an empty picture frame, prints selected image properties for the picture.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPictur

            With shpLoop.PictureFormat

                  If .IsEmpty = msoFalse Then

                       Debug.Print "File Name: " & .Filename
                       Debug.Print "Horizontal Scaling: " & .Horizo
                       Debug.Print "Vertical Scaling: " & .Vertical
                       Debug.Print "File size in publication: " & .

                  End If

            End With
        End If
    Next shpLoop
Next pgLoop
```


Show All

IsGreyScale Property

Returns a **MsoTriState** constant that indicates whether the picture is a greyscale image. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The picture is not a greyscale image.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified picture is a greyscale image.

expression.**IsGreyScale**()

expression Required. An expression that returns a **PictureFormat** object.

Example

The following example returns a list of the greyscale pictures contained in the active publication.

```
Sub ListGreyScalePictures()
Dim pgLoop As Page
Dim shpLoop As Shape

    For Each pgLoop In ActiveDocument.Pages
        For Each shpLoop In pgLoop.Shapes

            If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPi

                With shpLoop.PictureFormat
                    If .IsEmpty = msoFalse And .IsGreyScale = msoCTr

                        Debug.Print .Filename
                        Debug.Print "Page " & pgLoop.PageNumber

                    End If
                End With

            End If

        Next shpLoop
    Next pgLoop

End Sub
```

IsGroupMember Property

Returns **True** if the specified shape is a member of a group, **False** otherwise. Read-only **Boolean**.

expression.**IsGroupMember**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The object returned by the **ParentGroupShape** property can be used to determine the parent shape for the group.

Example

The following statement can be used to return a **True** value if the first shape of the active publication is a group member.

```
blnGrouped = Application.ActiveDocument.MasterPages _
    .Item.Shapes(1).IsGroupMember
```

IsHeader Property

True if the specified **HeaderFooter** object is a header, **False** if it is a footer. Read only **Boolean**.

expression.**IsHeader**

expression Required. An expression that returns a **HeaderFooter** object.

Example

The following example creates a new collection and fills it with the headers and footer from each master page. The collection is then iterated and a test is made to determine if the **HeaderFooter** object is a header or a footer, then appropriate text is written to the header or footer.

```
Dim objHeadersFooters As Collection
Dim objMasterPage As page
Dim objHeadFoot As HeaderFooter

Set objHeadersFooters = New Collection
For Each objMasterPage In ActiveDocument.masterPages
    objHeadersFooters.Add objMasterPage.Header
    objHeadersFooters.Add objMasterPage.Footer
Next objMasterPage

For Each objHeadFoot In objHeadersFooters
    If objHeadFoot.IsHeader = True Then
        objHeadFoot.TextRange.Text = "Header Text"
    ElseIf objHeadFoot.IsHeader = False Then
        objHeadFoot.TextRange.Text = "Footer Text"
    End If
Next
```

IsHorizontal Property

True if the specified **WebNavigationBarSet** has a horizontal orientation. Read-only **Boolean**.

expression.**IsHorizontal**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Remarks

This property is used to determine the orientation of the navigation bar set prior to setting certain properties that assume a horizontal orientation such as the **HorizontalAlignment** property.

Example

This example adds the first navigation bar in the **WebNavigationBarSets** collection of the active document to each page, and then sets the button style to **small**. A test is performed to determine whether the navigation bar set is horizontal or not. If it is not, the **ChangeOrientation** method is called and the orientation is set to **horizontal**. After the navigation bar is oriented horizontally, the horizontal button count is set to **3** and the horizontal alignment of the buttons is set to **left**.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets(1)
With objWebNav
    .AddToEveryPage Left:=10, Top:=10
    .ButtonStyle = pbnbButtonStyleSmall
    If .IsHorizontal = False Then
        .ChangeOrientation pbNavBarOrientHorizontal
    End If
    .HorizontalButtonCount = 3
    .HorizontalAlignment = pbnbAlignLeft
End With
```


Show All

IsInline Property

Returns an **MsoTriState** constant that specifies whether a shape is inline. Read-only.

MsoTriState can be one of these **MsoTriState** constants.

msoCTrue

msoFalse if a shape is not contained in a text run.

msoTriStateMixed

msoTriStateToggle

msoTrue if a shape is contained in a text run.

expression.**IsInline**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the first shape (a text frame) on the first page of the publication to see if it is inline. If it is not, a search is done within that shape to find any inline shapes within the text frame. Any inline shapes that are found have the **ForeColor** property set to red.

```
Dim theShape As Shape
Dim i As Integer

Set theShape = ActiveDocument.Pages(1).Shapes(1)

If Not theShape.IsInline = True Then
    With theShape.TextFrame.Story.TextRange
        If .InlineShapes.Count > 0 Then
            For i = 1 To .InlineShapes.Count
                .InlineShapes(i).Select
                .InlineShapes(i).Fill.ForeColor.RGB = vbRed
            Next
        End If
    End With
End If
```

IsLeading Property

True if the specified **Page** object is a leading page of a two page spread. Read only **Boolean**.

expression.**IsLeading**

expression Required. An expression that returns a **Page** object.

Example

The following example diplays for each page whether the page is a trailing or leading page in the publication.

```
Dim objPage As Page
Dim strPageInfo As String
For Each objPage In ActiveDocument.Pages
    strPageInfo = "Page number " & objPage.PageNumber
    If objPage.IsLeading Then
        strPageInfo = strPageInfo & " is a leading page." & Chr(13)
    ElseIf objPage.IsTrailing Then
        strPageInfo = strPageInfo & " is a trailing page." & Chr(13)
    End If
    MsgBox strPageInfo
Next objPage
```


Show All

IsLinked Property

Returns a **MsoTriState** constant indicating whether the specified picture is a linked picture or OLE object. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The picture is not a linked picture.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified picture is a linked picture.

expression.**IsLinked**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

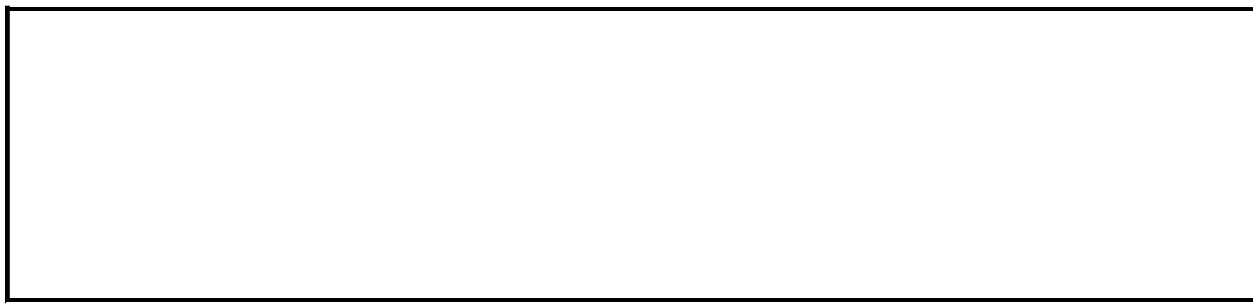
Returns **msoFalse** for pasted or embedded pictures and OLE objects.

If a picture or OLE object is linked, several additional properties of the **PictureFormat** object dealing with the original picture (such as **OriginalFileSize**) are accessible.

Example

The following example returns whether the first shape on the first page of the active publication contains an alpha channel. If the picture is linked, and the original picture contains an alpha channel, that is also returned. This example assumes the shape is a picture.

```
With ActiveDocument.Pages(1).Shapes(1).PictureFormat
    If .HasAlphaChannel = msoTrue Then
        Debug.Print .Filename
        Debug.Print "This picture contains an alpha channel."

        If .IsLinked = msoTrue Then
            If .OriginalHasAlphaChannel = msoTrue Then
                Debug.Print "The linked picture " & _
                "also contains an alpha channel."
            End If
        End If
    End If
End With
```



IsPostscriptPrinter Property

Returns **True** if the active printer is a PostScript printer. Read-only **Boolean**.

expression.**IsPostscriptPrinter**

expression Required. An expression that returns a **AdvancedPrintOptions** object.

Remarks

The following properties of the **AdvancedPrintOptions** object are only accessible if the active printer is a Postscript printer: **HorizontalFlip**, **VerticalFlip**, and **NegativeImage**.

Use the **ActivePrinter** property to specify the active printer for a publication.

Example

The following example determines if the active printer is a PostScript printer. If it is, the active publication is set to print as a horizontally and vertically mirrored, negative image of itself.

```
Sub PrepToPrintToFilmOnImagesetter()

With ActiveDocument.AdvancedPrintOptions
    If .IsPostscriptPrinter = True Then
        .HorizontalFlip = True
        .VerticalFlip = True
        .NegativeImage = True
    End If
End With

End Sub
```

IsTrailing Property

True if the specified **Page** object is a trailing page of a two page spread. Read only **Boolean**.

expression.**IsTrailing**

expression Required. An expression that returns a **Page** object.

Example

The following example diplays for each page whether the page is a trailing or leading page in the publication.

```
Dim objPage As Page
Dim strPageInfo As String
For Each objPage In ActiveDocument.Pages
    strPageInfo = "Page number " & objPage.PageNumber
    If objPage.IsLeading Then
        strPageInfo = strPageInfo & " is a leading page." & Chr(13)
    ElseIf objPage.IsTrailing Then
        strPageInfo = strPageInfo & " is a trailing page." & Chr(13)
    End If
    MsgBox strPageInfo
Next objPage
```


Show All

IsTrueColor Property

Returns an **MsoTriState** constant indicating whether the specified picture or OLE object contains color data of 24 bits per channel or greater. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The specified picture does not contain color data of 24 bits per channel or greater.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified picture contains color data of 24 bits per channel or greater.

expression.**IsTrueColor**()

expression Required. An expression that returns a **PictureFormat** object.

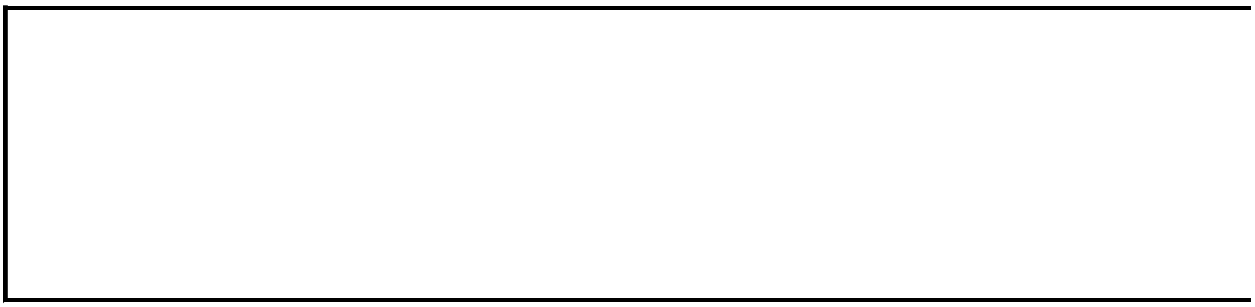
Remarks

For pictures that are not TrueColor, use the **ColorsInPalette** property of the **PictureFormat** object to determine the number of colors in the picture's palette.

Example

The following example tests each picture in the active document and prints out whether the picture is TrueColor, and if not prints how many colors are in the picture's palette.

```
For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Or shpLoop.Type = pbPictur

            With shpLoop.PictureFormat
                If .IsEmpty = msoFalse Then
                    Debug.Print .Filename
                    If .IsTrueColor = msoTrue Then
                        Debug.Print "This picture is TrueColor"
                    Else
                        Debug.Print "This picture contains " & .Colo
                    End If
                End If
            End With

        End If
    Next shpLoop
Next pgLoop
```



IsTwoPageMaster Property

True if the specified **Page** object is a two-page master. Read/write **Boolean**.

expression.**IsTwoPageMaster**

expression Required. An expression that returns a **Page** object from the **MasterPages** collection.

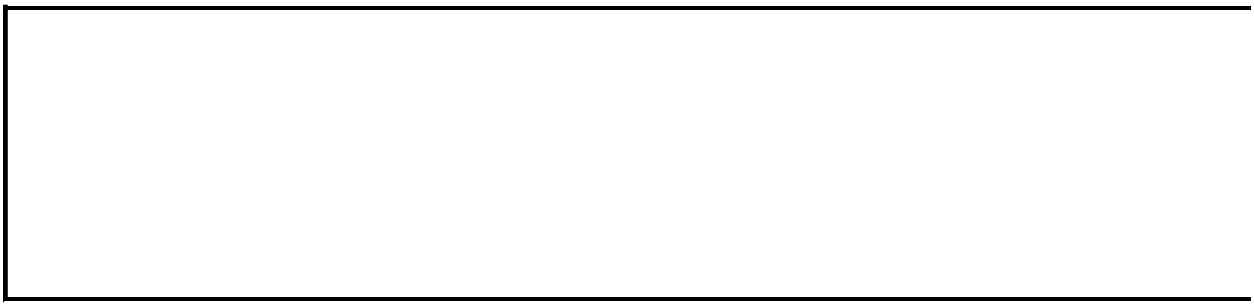
Remarks

This method works for master pages only. Returns a **This feature is only for master pages** error when attempting to access this property from a publication page object.

Example

The following example adds text to each header of a two-page master page specifying the master page **PageNumber** and its place in the spread: 1 or 2.

```
Dim objMasterPage As Page
Dim pageCount As Long
Dim i As Long
pageCount = ActiveDocument.MasterPages.Count
For i = 1 To pageCount
    Set objMasterPage = ActiveDocument.MasterPages(i)
    If objMasterPage.IsTwoPageMaster Then
        objMasterPage.Header.TextRange.Text = "MasterPage " & _
            objMasterPage.PageNumber & ", Page 1 of 2"
        i = i + 1
        Set objMasterPage = ActiveDocument.MasterPages(i)
        objMasterPage.Header.TextRange.Text = "MasterPage " & _
            objMasterPage.PageNumber & ", Page 2 of 2"
    End If
Next i
```



IsWizard Property

Returns **True** if the specified publication is a publication generated by a Publisher wizard. Read-only **Boolean**.

expression.**IsWizard**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **Wizard** property of the **Document** object to access the wizard for the specified publication.

Example

The following example tests to determine whether the active document is a wizard publication. If it is, certain wizard properties are returned.

```
With ActiveDocument
        If .IsWizard = True Then
            Debug.Print .Wizard.Name
            Debug.Print .Wizard.ID
        End If
End With
```

IsWizardPage Property

Returns **True** if the specified page is a Publisher wizard page. Read-only **Boolean**.

expression.**IsWizardPage**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Wizard pages are special page types for certain types of Publisher wizards (such as Newsletters, Catalogs, and Web Wizards) that can be inserted into a publication.

Use the **Wizard** property of the **Page** object to access the wizard for the specified page.

Example

The following example tests to determine whether the specified page is a wizard page. If it is, certain wizard properties are returned.

```
With ActiveDocument.Pages(1)
    If .IsWizardPage = True Then

        With .Wizard
            Debug.Print .Name
            Debug.Print .Properties(1).Name
            Debug.Print .Properties(1).CurrentValueId
        End With

    End If
End With
```


Show All

Italic Property

Returns or sets an **MsoTriState** constant indicating whether the specified text is formatted as italic. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters in the range are formatted as italic.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified text.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All of the characters in the range are formatted as italic.

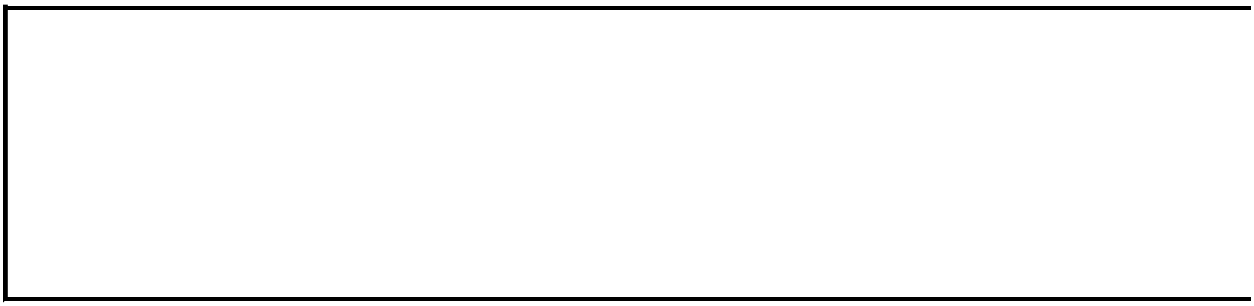
expression.**Italic**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests all the text in the second story of the active publication and if it has mixed italics, it sets all the text to italic. If the text is all italic or not italic, a message is dispalyed informing the user there are no mixed italics.

```
Sub ItalicStory()

    Dim stf As Font

    Set stf = Application.ActiveDocument.Stories(2).TextRange.Font
    With stf
        If .Italic = msoTriStateMixed Then
            .Italic = msoTrue
        Else
            MsgBox "There are no mixed italics in this story."
        End If
    End With

End Sub
```




Show All

ItalicBi Property

Returns or sets an **MsoTriState** constant indicating whether the specified text is formatted as italic; applies to text in a right-to-left language. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters in the range are formatted as italic.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified text.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All of the characters in the range are formatted as italic.

expression.**ItalicBi**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the text in the first story and displays one of two possible text boxes depending on if the text is right-to-left formatted and if its font is italicized.

```
Sub ItalicRtoL()

    Dim stf As Font

    Set stf = Application.ActiveDocument.Stories(1).TextRange.Font
    With stf
        If .ItalicBi = msoTrue Then
            MsgBox "This story is right-to-left and is italicized."
        Else
            MsgBox "This story is either not right-to-left" & _
                " or it is not italicized"
        End If
    End With

End Sub
```


Show All

Item Property

 Item property as it applies to the **Adjustments** object.

Returns or sets a **Variant** indicating the adjustment value specified by the ***Index*** argument. Read/write.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Integer**. The index number of the adjustment.

Remarks

AutoShapes, connectors, and WordArt objects can have up to eight adjustments.

For linear adjustments, an adjustment value of 0.0 generally corresponds to the left or top edge of the shape, and a value of 1.0 generally corresponds to the right or bottom edge of the shape. However, adjustments can pass beyond shape boundaries for some shapes. For radial adjustments, an adjustment value of 1.0 corresponds to the width of the shape. For angular adjustments, the adjustment value is specified in degrees.

The **Item** property applies only to shapes that have adjustments.

 Item property as it applies to the **ColorSchemes** object.

Returns the specified **ColorScheme** object from a **ColorSchemes** collection. Read-only.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Variant**. The color scheme to return. Can be either the name of the color scheme as a string or the corresponding **PbColorScheme** constant.

PbColorScheme can be one of these PbColorScheme constants.
pbColorSchemeAlpine
pbColorSchemeAqua
pbColorSchemeBerry
pbColorSchemeBlackGray
pbColorSchemeBlackWhite
pbColorSchemeBrown
pbColorSchemeBurgundy
pbColorSchemeCavern
pbColorSchemeCelebration
pbColorSchemeCherry

pbColorSchemeCitrus
pbColorSchemeClay
pbColorSchemeCranberry
pbColorSchemeCrocus
pbColorSchemeCustom
pbColorSchemeDarkBlue
pbColorSchemeDesert
pbColorSchemeField
pbColorSchemeFirstUserDefined
pbColorSchemeFjord
pbColorSchemeFloral
pbColorSchemeGarnet
pbColorSchemeGlacier
pbColorSchemeGreen
pbColorSchemeHeather
pbColorSchemeIris
pbColorSchemeIsland
pbColorSchemeIvy
pbColorSchemeLagoon
pbColorSchemeLilac
pbColorSchemeMahogany
pbColorSchemeMarine
pbColorSchemeMaroon
pbColorSchemeMeadow
pbColorSchemeMist
pbColorSchemeMistletoe
pbColorSchemeMonarch
pbColorSchemeMoss
pbColorSchemeMountain
pbColorSchemeMulberry
pbColorSchemeNavy
pbColorSchemeNutmeg
pbColorSchemeOcean

pbColorSchemeOlive
pbColorSchemeOrange
pbColorSchemeOrchid
pbColorSchemeParrot
pbColorSchemePeach
pbColorSchemePebbles
pbColorSchemePrairie
pbColorSchemePurple
pbColorSchemeRainForest
pbColorSchemeRed
pbColorSchemeRedwood
pbColorSchemeReef
pbColorSchemeSagebrush
pbColorSchemeSapphire
pbColorSchemeShamrock
pbColorSchemeSienna
pbColorSchemeSpice
pbColorSchemeSunrise
pbColorSchemeSunset
pbColorSchemeTeal
pbColorSchemeTidepool
pbColorSchemeTropics
pbColorSchemeTrout
pbColorSchemeVineyard
pbColorSchemeWaterfall
pbColorSchemeWildflower

 Item property as it applies to the **MasterPages** and **Pages** objects.

Returns the specified **Page** object from a **Pages** or **MasterPages** collection. Read-only.

expression.**Item**(***Item***)

expression Required. An expression that returns one of the above objects.

Item Required **Long**. The number of the page to return. For **MasterPages** collections, ***Item*** can either be 1 or 2 for the left and right master pages, respectively. For **Pages** collections, ***Item*** corresponds to a **Page** object's **PageIndex** property.

 Item property as it applies to all the other objects in the Applies to list.

Returns an individual object from a specified collection. Read-only.

expression.**Item**(***Index***)

expression Required. An expression that returns one of the objects in the Applies to list.

Index Required **Long**. The number of the object to return.

Example

 As it applies to the **Adjustments** object.

This example adds two crosses to the active publication and then sets the value for adjustment one (the only one for this type of AutoShape) on each cross.

```
With ActiveDocument.Pages(1).Shapes
    .AddShape(Type:=msoShapeCross, Left:=10, Top:=10, Width:=100, _
        Height:=100).Adjustments.Item(1) = 0.4
    .AddShape(Type:=msoShapeCross, Left:=150, Top:=10, Width:=100, _
        Height:=100).Adjustments.Item(1) = 0.2
End With
```

This example has the same result as the previous example even though it doesn't explicitly use the **Item** property.

```
With ActiveDocument.Pages(1).Shapes
    .AddShape(Type:=msoShapeCross, Left:=10, Top:=10, Width:=100, _
        Height:=100).Adjustments(1) = 0.4
    .AddShape(Type:=msoShapeCross, Left:=150, Top:=10, Width:=100, _
        Height:=100).Adjustments(1) = 0.2
End With
```

 As it applies to the **ColorSchemes** object.

This example sets the color scheme of the active publication to the Aqua color scheme.

```
ActiveDocument.ColorScheme = ColorSchemes.Item(Index:=pbColorSchemeA
```

 As it applies to the **Hyperlinks** object.

This example displays the address of the first hyperlink in shape one of the active publication.

```
MsgBox "Address of first hyperlink: " _
    & ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Hyperlinks.Item(1).Address
```

 As it applies to the **MasterPages** and **Pages** object.

This example displays the page number, page index, and page ID of the first page in the active publication.

```
With ActiveDocument.Pages.Item(1)
    Debug.Print "Page number = " & .PageNumber
    Debug.Print "Page index = " & .PageIndex
    Debug.Print "Page ID = " & .PageID
End With
```

 As it applies to the **Plates** object.

This example displays the name of the first color plate in the active publication.

```
MsgBox "Name of first color plate: " _
    & ActiveDocument.Plates.Item(1).Name
```

 As it applies to the **RulerGuides** object.

This example sets the position of the first ruler guide to 3 inches from the edge of the publication.

```
ActiveDocument.Pages(1).RulerGuides _
    .Item(1).Position = InchesToPoints(3)
```

KashidaPercentage Property

Returns or sets a **Long** indicating the percentage by which kashidas are to be lengthened for the specified paragraphs. Valid values are from 0 to 100. Read/write.

expression.**KashidaPercentage**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **Alignment** property of the specified paragraphs must be set to **pbParagraphAlignmentKashida** or the **KashidaPercentage** property is ignored.

Example

The following example sets the paragraphs in shape one on page one of the active publication to kashida alignment and specifies that kashidas are to be lengthened by 20 percent.

```
With ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.ParagraphFormat
    .Alignment = pbParagraphAlignmentKashida
    .KashidaPercentage = 20
End With
```


Show All

KeepLinesTogether Property

Sets or returns an **msoTriState** that represents whether or not all lines in the specified paragraph will remain in the same text box. Read/write.

msoCTrue
msoFalse All lines will remain in the same text box.
msoTriStateMixed
msoTriStateToggle
msoTrue All lines may not remain in the same text box.

expression.**KeepLinesTogether**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

This option ensures that there is not a text frame or column break between the lines of the specified paragraph. If the paragraphs are too large for the text frame or column, the first line will start at the top of the next text frame or column.

The default setting for this property is **msoFalse**.

Example

This example sets the **KeepLinesTogether** property to **msoTrue** for the specified **ParagraphFormat** object.

```
Dim objParaForm As ParagraphFormat
Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Paragraphs(1).ParagraphFormat
objParaForm.KeepLinesTogether = msoTrue
```


Show All

KeepWithNext Property

Sets or returns an **msoTriState** that represents whether or not the following paragraph will remain in the same text box as the specified paragraph. Read/write.

msoCTrue
msoFalse Next paragraph will remain in the same text box.
msoTriStateMixed
msoTriStateToggle
msoTrue Next paragraph may not remain in the same text box.

expression.**KeepWithNext**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

The purpose of keep with next is to prevent hanging headings in a document. To do this a user may set this property to **msoTrue** for all headings.

The default setting for this property is **msoFalse**.

Example

This example sets the **KeepWithNext** property to **msoTrue** for the specified **ParagraphFormat** object.

```
Dim objParaForm As ParagraphFormat
Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Paragraphs(1).ParagraphFormat
objParaForm.KeepWithNext = msoTrue
```


Show All

KernedPairs Property

Sets or returns an **MsoTriState** constant that indicates that character pairs in a WordArt object have been kerned. Read/write.

MsoTriState can be one of these MsoTriState constants.
msoCTrue Not used with this property.
msoFalse Character pairs in the specified WordArt object have not been kerned.
msoTriStateMixed Not used with this property.
msoTriStateToggle Toggles between **msoTrue** and **msoFalse**.
msoTrue Character pairs in the specified WordArt object have been kerned.

expression.**KernedPairs**

expression Required. An expression that returns a **TextEffectFormat** object.

Example

This example turns on character pair kerning for all WordArt objects in the active publication.

```
Sub KernedWordArt()
    Dim pagPage As Page
    Dim shpShape As Shape
    For Each pagPage In ActiveDocument.Pages
        For Each shpShape In pagPage.Shapes
            If shpShape.Type = msoTextEffect Then
                shpShape.TextEffect.KernedPairs = msoTrue
            End If
        Next
    Next
End Sub
```

Kerning Property

Returns or sets a **Variant** indicating the amount of horizontal spacing Microsoft Publisher applies to the characters in the text range. Read/write.

expression.**Kerning**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

When setting this property, numeric values are considered to be in points, and **String** values may be in any unit supported by Publisher. Return values are of type **Single** and in points. Negative values bring characters closer together than normal, and positive values spread characters farther apart than normal. The valid range is -600.0 to 600.0 points.

Use the **InchesToPoints** method to convert inches to points.

Example

This example adjusts the kerning of all text in the first story to 6 point.

```
Application.ActiveDocument.Stories(1).TextRange.Font.Kerning = 6
```

Keywords Property

Returns or sets a **String** that represents the keywords for a Web page within a Web publication. Read/write.

expression.**Keywords**

expression Required. An expression that returns a **WebPageOptions** object.

Example

The following example sets the keywords for page four of the active publication.

```
Dim theWPO As WebPageOptions

Set theWPO = ActiveDocument.Pages(4).WebPageOptions

With theWPO
    .Keywords = "software, hardware, computers"
End With
```

Label Property

Returns or sets a **Label** object that represents a single unique label design available on the system. Read/write.

expression.**Label**

expression Required. An expression that returns a **PageSetup** object.

Remarks

The returned **Label** object is contained within the AvailableLabels collection, which is accessed by using the **AvailableLabels** property.

Only labels that are relevant in the current language/locale are available programmatically.

Example

The following example demonstrates using the **Label** property to return the fifth label available on the system and modify its properties to create a custom label. Various label properties are set for this label, and then a text box and some text are added to the label. All pages that contain this custom label will have the properties that are set in this example.

```
Dim theLabel As Label

With ActiveDocument.PageSetup
    .Label = .AvailableLabels(5)  ' Label 5 is Avery 5164
    Set theLabel = .Label
    With theLabel
        .LeftMargin = InchesToPoints(0.15)
        .TopMargin = InchesToPoints(0.15)
        .HorizontalGap = InchesToPoints(0.1)
        .VerticalGap = InchesToPoints(0.1)
    End With
End With

With ActiveDocument.Pages(4).Shapes.AddShape(msoShapeRectangle, _
        5, 5, (theLabel.Width - 10), (theLabel.Height - 10))
    With .TextFrame.TextRange
        .Font.Name = "Verdana"
        .Font.Size = 12
        .Text = "Here is some label text."
    End With
End With
```

The following example demonstrates that certain properties of the **Label** object are read-only if accessed without using **.PageSetup.Label**.

```
Dim theLabel As Label

Set theLabel = ActiveDocument.PageSetup.AvailableLabels(5)

With theLabel
    ' Trying to set any of the following four properties will return
    ' All of these properties are read-only if accessed without usin
    ' .PageSetup.Label.
    .LeftMargin = InchesToPoints(0.15)
    .TopMargin = InchesToPoints(0.15)
    .HorizontalGap = InchesToPoints(0.1)
```

```
    .VerticalGap = InchesToPoints(0.1)
End With
```


Show All

Language Property

Returns a **Long** that represents the language selected for the Microsoft Publisher user interface. Read-only.

expression.**Language**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **Language** property can return any **MsoLanguageID** constant.

MsoLanguageID can be one of these MsoLanguageID constants.
msoLanguageIDAfrikaans
msoLanguageIDAlbanian
msoLanguageIDAmharic
msoLanguageIDArabic
msoLanguageIDArabicAlgeria
msoLanguageIDArabicBahrain
msoLanguageIDArabicEgypt
msoLanguageIDArabicIraq
msoLanguageIDArabicJordan
msoLanguageIDArabicKuwait
msoLanguageIDArabicLebanon
msoLanguageIDArabicLibya
msoLanguageIDArabicMorocco
msoLanguageIDArabicOman
msoLanguageIDArabicQatar
msoLanguageIDArabicSyria
msoLanguageIDArabicTunisia
msoLanguageIDArabicUAE
msoLanguageIDArabicYemen
msoLanguageIDArmenian
msoLanguageIDAssamese
msoLanguageIDAzeriCyrillic
msoLanguageIDAzeriLatin
msoLanguageIDBasque
msoLanguageIDBelgianDutch
msoLanguageIDBelgianFrench
msoLanguageIDBengali

msoLanguageIDBrazilianPortuguese
msoLanguageIDBulgarian
msoLanguageIDBurmese
msoLanguageIDByelorussian
msoLanguageIDCatalan
msoLanguageIDCherokee
msoLanguageIDChineseHongKong
msoLanguageIDChineseMacao
msoLanguageIDChineseSingapore
msoLanguageIDCroatian
msoLanguageIDCzech
msoLanguageIDDanish
msoLanguageIDDutch
msoLanguageIDEnglishAUS
msoLanguageIDEnglishBelize
msoLanguageIDEnglishCanadian
msoLanguageIDEnglishCaribbean
msoLanguageIDEnglishIreland
msoLanguageIDEnglishJamaica
msoLanguageIDEnglishNewZealand
msoLanguageIDEnglishPhilippines
msoLanguageIDEnglishSouthAfrica
msoLanguageIDEnglishTrinidad
msoLanguageIDEnglishUK
msoLanguageIDEnglishUS
msoLanguageIDEnglishZimbabwe
msoLanguageIDEstonian
msoLanguageIDFaeroese
msoLanguageIDFarsi
msoLanguageIDFinnish
msoLanguageIDFrench
msoLanguageIDFrenchCameroon
msoLanguageIDFrenchCanadian

msoLanguageIDFrenchCotedIvoire
msoLanguageIDFrenchLuxembourg
msoLanguageIDFrenchMali
msoLanguageIDFrenchMonaco
msoLanguageIDFrenchReunion
msoLanguageIDFrenchSenegal
msoLanguageIDFrenchWestIndies
msoLanguageIDFrenchZaire
msoLanguageIDFrisianNetherlands
msoLanguageIDGaelicIreland
msoLanguageIDGaelicScotland
msoLanguageIDGalician
msoLanguageIDGeorgian
msoLanguageIDGerman
msoLanguageIDGermanAustria
msoLanguageIDGermanLiechtenstein
msoLanguageIDGermanLuxembourg
msoLanguageIDGreek
msoLanguageIDGujarati
msoLanguageIDHebrew
msoLanguageIDHindi
msoLanguageIDHungarian
msoLanguageIDIcelandic
msoLanguageIDIndonesian
msoLanguageIDInuktitut
msoLanguageIDItalian
msoLanguageIDJapanese
msoLanguageIDKannada
msoLanguageIDKashmiri
msoLanguageIDKazakh
msoLanguageIDKhmer
msoLanguageIDKirghiz
msoLanguageIDKonkani

msoLanguageIDKorean
msoLanguageIDLao
msoLanguageIDLatvian
msoLanguageIDLithuanian
msoLanguageIDMacedonian
msoLanguageIDMalayalam
msoLanguageIDMalayBruneiDarussalam
msoLanguageIDMalaysian
msoLanguageIDMaltese
msoLanguageIDManipuri
msoLanguageIDMarathi
msoLanguageIDMexicanSpanish
msoLanguageIDMixed
msoLanguageIDMongolian
msoLanguageIDNepali
msoLanguageIDNone
msoLanguageIDNoProofing
msoLanguageIDNorwegianBokmol
msoLanguageIDNorwegianNynorsk
msoLanguageIDOriya
msoLanguageIDOromo
msoLanguageIDPolish
msoLanguageIDPortuguese
msoLanguageIDPunjabi
msoLanguageIDRhaetoRomanic
msoLanguageIDRomanian
msoLanguageIDRomanianMoldova
msoLanguageIDRussian
msoLanguageIDRussianMoldova
msoLanguageIDSamiLappish
msoLanguageIDSanskrit
msoLanguageIDSerbianCyrillic
msoLanguageIDSerbianLatin

msoLanguageIDSesotho
msoLanguageIDSimplifiedChinese
msoLanguageIDSindhi
msoLanguageIDSlovak
msoLanguageIDSlovenian
msoLanguageIDSorbian
msoLanguageIDSpanish
msoLanguageIDSpanishArgentina
msoLanguageIDSpanishBolivia
msoLanguageIDSpanishChile
msoLanguageIDSpanishColombia
msoLanguageIDSpanishCostaRica
msoLanguageIDSpanishDominicanRepublic
msoLanguageIDSpanishEcuador
msoLanguageIDSpanishElSalvador
msoLanguageIDSpanishGuatemala
msoLanguageIDSpanishHonduras
msoLanguageIDSpanishModernSort
msoLanguageIDSpanishNicaragua
msoLanguageIDSpanishPanama
msoLanguageIDSpanishParaguay
msoLanguageIDSpanishPeru
msoLanguageIDSpanishPuertoRico
msoLanguageIDSpanishUruguay
msoLanguageIDSpanishVenezuela
msoLanguageIDSutu
msoLanguageIDSwahili
msoLanguageIDSwedish
msoLanguageIDSwedishFinland
msoLanguageIDSwissFrench
msoLanguageIDSwissGerman
msoLanguageIDSwissItalian
msoLanguageIDTajik

msoLanguageIDTamil
msoLanguageIDTatar
msoLanguageIDTelugu
msoLanguageIDThai
msoLanguageIDTibetan
msoLanguageIDTigrignaEritrea
msoLanguageIDTigrignaEthiopic
msoLanguageIDTraditionalChinese
msoLanguageIDTsonga
msoLanguageIDTswana
msoLanguageIDTurkish
msoLanguageIDTurkmen
msoLanguageIDUkrainian
msoLanguageIDUrdu
msoLanguageIDUzbekCyrillic
msoLanguageIDUzbekLatin
msoLanguageIDVenda
msoLanguageIDVietnamese
msoLanguageIDWelsh
msoLanguageIDXhosa
msoLanguageIDZulu

Example

This example displays a message stating whether the language selected for the Microsoft Publisher user interface is U.S. English.

```
Sub LangSetting()
    If Application.Language = msoLanguageIDEnglishUS Then
        MsgBox "The user interface language is U.S. English."
    Else
        MsgBox "The user interface language is not U.S. English."
    End If
End Sub
```


Show All

LanguageID Property

Returns or sets a **MsoLanguageID** constant that represents the language for the specified object. Read/write.

MsoLanguageID can be one of these MsoLanguageID constants.

msoLanguageIDAfrikaans
msoLanguageIDAlbanian
msoLanguageIDAmharic
msoLanguageIDArabic
msoLanguageIDArabicAlgeria
msoLanguageIDArabicBahrain
msoLanguageIDArabicEgypt
msoLanguageIDArabicIraq
msoLanguageIDArabicJordan
msoLanguageIDArabicKuwait
msoLanguageIDArabicLebanon
msoLanguageIDArabicLibya
msoLanguageIDArabicMorocco
msoLanguageIDArabicOman
msoLanguageIDArabicQatar
msoLanguageIDArabicSyria
msoLanguageIDArabicTunisia
msoLanguageIDArabicUAE
msoLanguageIDArabicYemen
msoLanguageIDArmenian
msoLanguageIDAssamese
msoLanguageIDAzeriCyrillic
msoLanguageIDAzeriLatin
msoLanguageIDBasque
msoLanguageIDBelgianDutch
msoLanguageIDBelgianFrench

msoLanguageIDBengali
msoLanguageIDBrazilianPortuguese
msoLanguageIDBulgarian
msoLanguageIDBurmese
msoLanguageIDByelorussian
msoLanguageIDCatalan
msoLanguageIDCherokee
msoLanguageIDChineseHongKong
msoLanguageIDChineseMacao
msoLanguageIDChineseSingapore
msoLanguageIDCroatian
msoLanguageIDCzech
msoLanguageIDDanish
msoLanguageIDDivehi
msoLanguageIDDutch
msoLanguageIDEdo
msoLanguageIDEnglishAUS
msoLanguageIDEnglishBelize
msoLanguageIDEnglishCanadian
msoLanguageIDEnglishCaribbean
msoLanguageIDEnglishIreland
msoLanguageIDEnglishJamaica
msoLanguageIDEnglishNewZealand
msoLanguageIDEnglishPhilippines
msoLanguageIDEnglishSouthAfrica
msoLanguageIDEnglishTrinidad
msoLanguageIDEnglishUK
msoLanguageIDEnglishUS
msoLanguageIDEnglishZimbabwe
msoLanguageIDEstonian
msoLanguageIDFaeroese
msoLanguageIDFarsi
msoLanguageIDFilipino

msoLanguageIDFinnish
msoLanguageIDFrench
msoLanguageIDFrenchCameroon
msoLanguageIDFrenchCanadian
msoLanguageIDFrenchCotedIvoire
msoLanguageIDFrenchLuxembourg
msoLanguageIDFrenchMali
msoLanguageIDFrenchMonaco
msoLanguageIDFrenchReunion
msoLanguageIDFrenchSenegal
msoLanguageIDFrenchWestIndies
msoLanguageIDFrenchZaire
msoLanguageIDFrisianNetherlands
msoLanguageIDFulfulde
msoLanguageIDGaelicIreland
msoLanguageIDGaelicScotland
msoLanguageIDGalician
msoLanguageIDGeorgian
msoLanguageIDGerman
msoLanguageIDGermanAustria
msoLanguageIDGermanLiechtenstein
msoLanguageIDGermanLuxembourg
msoLanguageIDGreek
msoLanguageIDGuarani
msoLanguageIDGujarati
msoLanguageIDHausa
msoLanguageIDHawaiian
msoLanguageIDHebrew
msoLanguageIDHindi
msoLanguageIDHungarian
msoLanguageIDIbibio
msoLanguageIDIcelandic
msoLanguageIDIgbo

msoLanguageIDIndonesian
msoLanguageIDInuktitut
msoLanguageIDItalian
msoLanguageIDJapanese
msoLanguageIDKannada
msoLanguageIDKanuri
msoLanguageIDKashmiri
msoLanguageIDKazakh
msoLanguageIDKhmer
msoLanguageIDKirghiz
msoLanguageIDKonkani
msoLanguageIDKorean
msoLanguageIDKyrgyz
msoLanguageIDLao
msoLanguageIDLatin
msoLanguageIDLatvian
msoLanguageIDLithuanian
msoLanguageIDMacedonian
msoLanguageIDMalayalam
msoLanguageIDMalayBruneiDarussalam
msoLanguageIDMalaysian
msoLanguageIDMaltese
msoLanguageIDManipuri
msoLanguageIDMarathi
msoLanguageIDMexicanSpanish
msoLanguageIDMixed
msoLanguageIDMongolian
msoLanguageIDNepali
msoLanguageIDNone
msoLanguageIDNoProofing
msoLanguageIDNorwegianBokmol
msoLanguageIDNorwegianNynorsk
msoLanguageIDOriya

msoLanguageIDOromo
msoLanguageIDPashto
msoLanguageIDPolish
msoLanguageIDPortuguese
msoLanguageIDPunjabi
msoLanguageIDRhaetoRomanic
msoLanguageIDRomanian
msoLanguageIDRomanianMoldova
msoLanguageIDRussian
msoLanguageIDRussianMoldova
msoLanguageIDSamiLappish
msoLanguageIDSanskrit
msoLanguageIDSerbianCyrillic
msoLanguageIDSerbianLatin
msoLanguageIDSesotho
msoLanguageIDSimplifiedChinese
msoLanguageIDSindhi
msoLanguageIDSindhiPakistan
msoLanguageIDSinhalese
msoLanguageIDSlovak
msoLanguageIDSlovenian
msoLanguageIDSomali
msoLanguageIDSorbian
msoLanguageIDSpanish
msoLanguageIDSpanishArgentina
msoLanguageIDSpanishBolivia
msoLanguageIDSpanishChile
msoLanguageIDSpanishColombia
msoLanguageIDSpanishCostaRica
msoLanguageIDSpanishDominicanRepublic
msoLanguageIDSpanishEcuador
msoLanguageIDSpanishElSalvador
msoLanguageIDSpanishGuatemala

msoLanguageIDSpanishHonduras
msoLanguageIDSpanishModernSort
msoLanguageIDSpanishNicaragua
msoLanguageIDSpanishPanama
msoLanguageIDSpanishParaguay
msoLanguageIDSpanishPeru
msoLanguageIDSpanishPuertoRico
msoLanguageIDSpanishUruguay
msoLanguageIDSpanishVenezuela
msoLanguageIDSutu
msoLanguageIDSwahili
msoLanguageIDSwedish
msoLanguageIDSwedishFinland
msoLanguageIDSwissFrench
msoLanguageIDSwissGerman
msoLanguageIDSwissItalian
msoLanguageIDSyriac
msoLanguageIDTajik
msoLanguageIDTamazight
msoLanguageIDTamazightLatin
msoLanguageIDTamil
msoLanguageIDTatar
msoLanguageIDTelugu
msoLanguageIDThai
msoLanguageIDTibetan
msoLanguageIDTigrignaEritrea
msoLanguageIDTigrignaEthiopic
msoLanguageIDTraditionalChinese
msoLanguageIDTsonga
msoLanguageIDTswana
msoLanguageIDTurkish
msoLanguageIDTurkmen
msoLanguageIDUkrainian

msoLanguageIDUrdu
msoLanguageIDUzbekCyrillic
msoLanguageIDUzbekLatin
msoLanguageIDVenda
msoLanguageIDVietnamese
msoLanguageIDWelsh
msoLanguageIDXhosa
msoLanguageIDYi
msoLanguageIDYiddish
msoLanguageIDYoruba
msoLanguageIDZulu

expression.**LanguageID**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example formats as French the specified selection. This example assumes that the insertion point is in a text box.

```
Sub SetLanguage()
    Selection.TextRange.LanguageID = msoLanguageIDFrench
End Sub
```


Show All

LastRecord Property

Returns or sets a **Long** that represents the number of the last data record to be merged in a mail merge or catalog merge operation. Read/write.

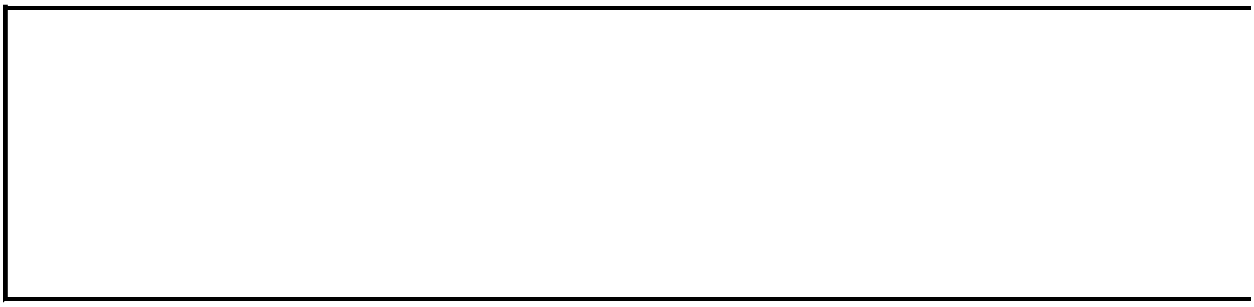
expression.**LastRecord**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example sets the active record as the first record to be merged and then sets the last record as the record two records forward in the data source. This example assumes that the active publication is a mail merge publication.

```
Sub RecordOne()
    With ActiveDocument.MailMerge
        .DataSource.FirstRecord = .DataSource.ActiveRecord
        .DataSource.LastRecord = .DataSource.ActiveRecord + 2
        .Execute Pause:=True
    End With
End Sub
```



LayoutGuides Property

Returns a **LayoutGuides** object consisting of the margin and grid layout guides for all pages including master pages in the publication.

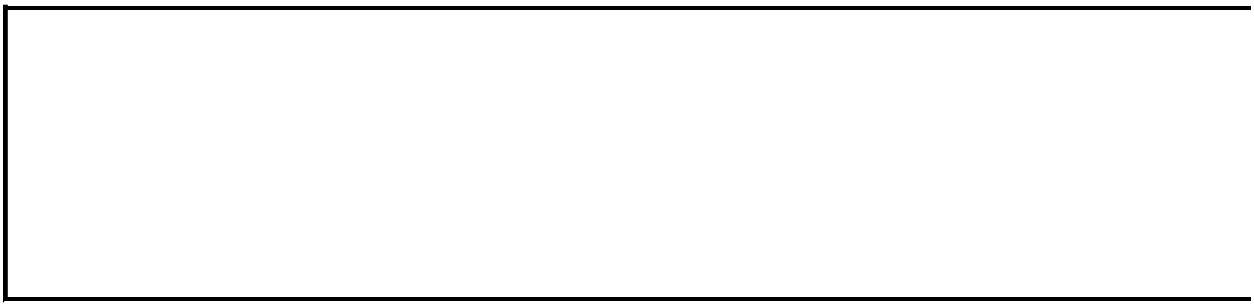
expression.**LayoutGuides**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example changes the grid layout guides so that there are three columns and five rows.

```
Dim layTemp As LayoutGuides

Set layTemp = ActiveDocument.LayoutGuides

With layTemp
    .Rows = 5
    .Columns = 3
End With
```




Show All

Leader Property

Sets or returns a **PbTabLeaderType** constant that represents the leader character for a tab stop. Read/write.

PbTabLeaderType can be one of these PbTabLeaderType constants.
pbTabLeaderBullet
pbTabLeaderDashes
pbTabLeaderDot
pbTabLeaderLine
pbTabLeaderNone

expression.**Leader**

expression Required. An expression that returns one of the objects in the Applies To list.

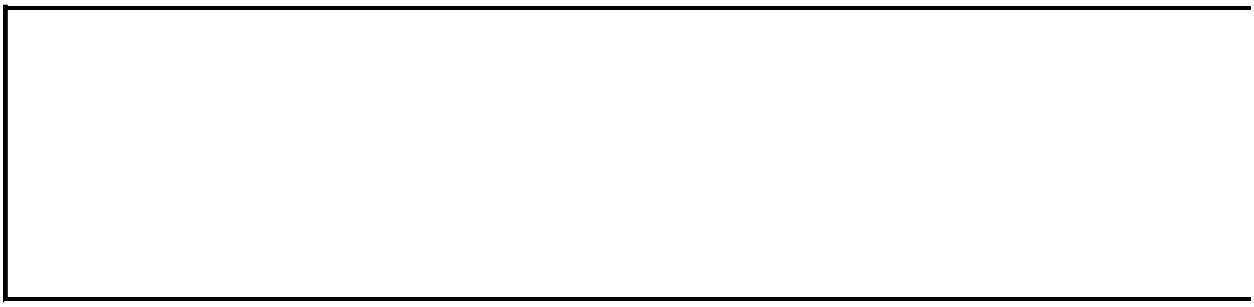
Example

This example changes the leader tab character of the selected paragraphs to dashes. This example assumes that the selected paragraph contains at least one tab stop.

```
Sub SetLeaderTab()
   Selection.TextRange.ParagraphFormat _
    .Tabs(1).Leader = pbTabLeaderDashes
End Sub
```

This example changes the leader tab character of the first paragraph in the specified text range to an underline. This example assumes that the specified paragraph contains at least one tab stop.

```
Sub SetNewTabLeader()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange.Paragraphs
        .ParagraphFormat.Tabs(1).Leader = pbTabLeaderLine
End Sub
```




Show All

Left Property

Left property as it applies to the **ReaderSpread** object.

Returns a **Single** indicating the position (in points) of the left edge of the reader spread from the workspace. Read-only.

expression.**Left**

expression Required. An expression that returns one of the above objects.

Left property as it applies to the **PrintableRect** object.

Returns a **Single** indicating the distance (in points) from the left edge of the printer sheet to the left edge of the printable rectangle. Read-only.

expression.**Left**

expression Required. An expression that returns one of the above objects.

Left property as it applies to the **Shape** object.

Returns or sets a **Variant** indicating the distance from the left edge of the page to the leftmost edge of the specified shape. Numeric values are in points; all other values are in any measurement supported by Publisher (for example, "2.5 in"). Read/write.

expression.**Left**

expression Required. An expression that returns one of the above objects.

Left property as it applies to the **ShapeRange** object.

Returns a **Variant** indicating the distance from the left edge of the page to the leftmost edge of all the shapes in the specified shape range. Numeric values are in points; all other values are in any measurement supported by Publisher (for example, "2.5 in"). Read-only.

expression.**Left**

expression Required. An expression that returns one of the above objects.

 **Left** property as it applies to the **Window** object.

Returns or sets a **Long** indicating the position (in points) of the left edge of the application window relative to the left edge of the screen. Read/write.

expression.**Left**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **Shape** object.

This example sets the horizontal position of the first shape in the active publication to 1 inch from the left edge of the page.

```
With ActiveDocument.Pages(1).Shapes(1)
    .Left = InchesToPoints(1)
End With
```

As it applies to the **Window** object.

This example sets the horizontal position of the active window to 100 points.

```
With ActiveDocument.ActiveWindow
    .WindowState = pbWindowStateNormal
    .Left = 100
    .Top = 0
End With
```

LeftIndent Property

Returns or sets a **Variant** that represents the left indent value (in points) for the specified paragraphs. Read/write.

expression.**LeftIndent**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example indents one-half inch the paragraph at the cursor position. This example assumes the insertion point is in a text box.

```
Sub IndentParagraph()
    Selection.TextRange.ParagraphFormat.LeftIndent = 36
End Sub
```

LeftMargin Property

Returns or sets a **Variant** that represents the distance (in points) between the left edge of the printer sheet and the left edge of the publication pages. Read/write.

expression.**LeftMargin**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

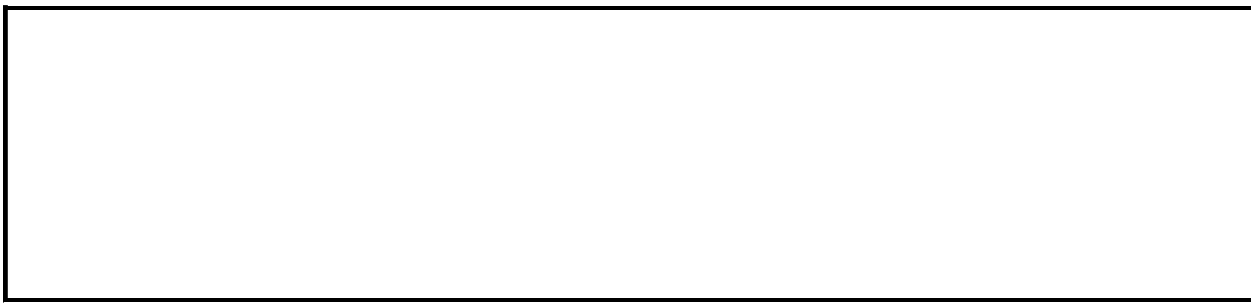
You can only use the **LeftMargin** property when printing multiple pages on a single sheet of printer paper.

When used with the **Label** object, the **LeftMargin** property is read/write only when accessed from **.PageSetup.Label**. Otherwise, it is read-only.

Example

This example specifies a width of a quarter of an inch for the area between the edge of the printer paper and the left edge of the pages in the active publication.

```
Sub SetLeftMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```




Show All

Length Property

 Length property as it applies to the **CalloutFormat** object.

Returns a **Variant** indicating the length (in points) of the first segment of the callout line (the segment attached to the text callout box) if the **AutoLength** property of the specified callout is set to **False**. Otherwise, an error occurs. Read-only.

expression.Length

expression Required. An expression that returns a **CalloutFormat** object.

Remarks

This property applies only to callouts whose lines consist of more than one segment (types **msoCalloutThree** and **msoCalloutFour**).

Use the **CustomLength** method to set the value of this property.

 Length property as it applies to the **TextRange** object.

Returns a **Long** value indicating the length of the specified text range, in characters. Read-only.

expression.Length

expression Required. An expression that returns a **TextRange** object.

Example

As it applies to the **CalloutFormat** object.

If the first line segment in the callout named co1 has a fixed length, this example specifies that the length of the first line segment in the callout named co2 will also be fixed at that length. For the example to work, both callouts must have multiple-segment lines.

```
Dim len1 As Single

With ActiveDocument.Pages(1).Shapes
    With .Item("co1").Callout
        If Not .AutoLength Then len1 = .Length
    End With
    If len1 Then .Item("co2").Callout _
        .CustomLength Length:=len1
End With
```

As it applies to the **TextRange** object.

This example sets the font size of a text frame on page two to 48 points if the text frame contains more than five characters, or it sets the font size to 72 points if the text frame has five or fewer characters.

```
With ActiveDocument.Pages(2).Shapes(1) _
        .TextFrame.TextRange
    If .Length > 5 Then
        .Font.Size = 48
    Else
        .Font.Size = 72
    End If
End With
```

Limit Property

Returns or sets a **Long** that represents the maximum number of characters that can be entered into a Web text box control. Read/write.

expression.**Limit**

expression Required. An expression that returns one of the objects in the Applies To list.

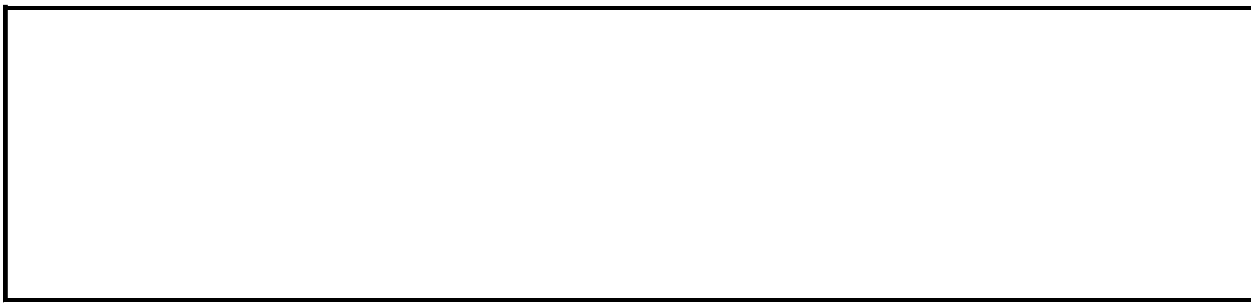
Remarks

Text box limits can be any number from 1 to 255 characters. Numbers higher than 255 will generate an error.

Example

This example creates a new Web text box control in the active publication, sets the default text and the character limit for the text box, and specifies that it is a required control.

```
Sub AddWebTextBoxControl()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlMultiLineTextBox, Left:=72, _
            Top:=72, Width:=300, Height:=100).WebTextBox
        .DefaultText = "Please enter text here."
        .Limit = 200
        .RequiredControl = msoTrue
    End With
End Sub
```



Line Property

Returns a **LineFormat** object that contains line formatting properties for the specified shape. (For a line, the **LineFormat** object represents the line itself; for a shape with a border, the **LineFormat** object represents the border.)

expression.**Line**

expression Required. An expression that returns one of the objects in the Applies To list.

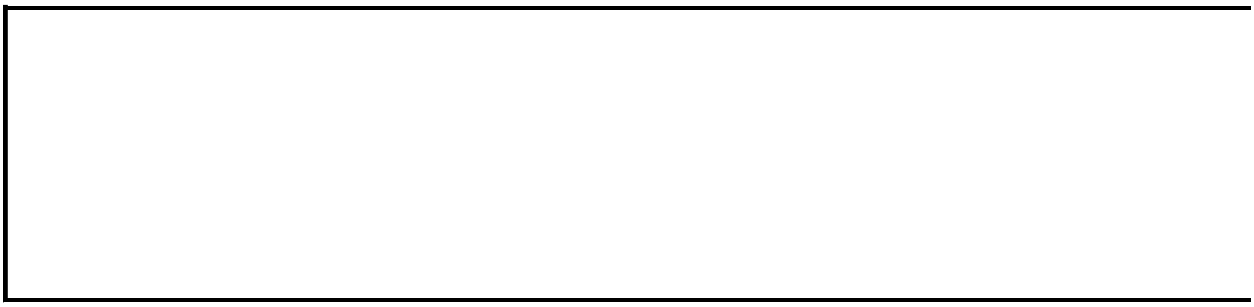
Example

This example adds a blue dashed line to the active publication.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=10, BeginY:=10, _
        EndX:=250, EndY:=250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

This example adds a cross to the first page and then sets its border to be 8 points thick and red.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeCross, _
        Left:=10, Top:=10, Width:=50, Height:=70).Line
    .Weight = 8
    .ForeColor.RGB = RGB(255, 0, 0)
End With
```



LineSpacing Property

Returns or sets a **Variant** that represents the line spacing (in number of lines) for the specified paragraphs. Read/write.

expression.**LineSpacing**

expression Required. An expression that returns one of the objects in the Applies To list.

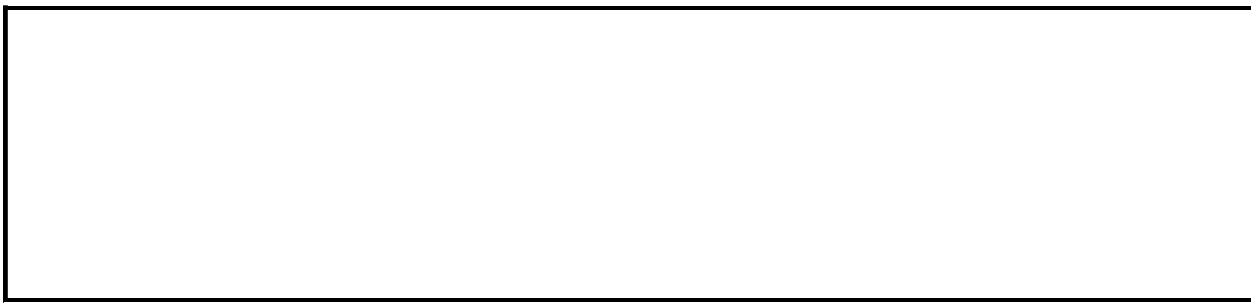
Remarks

You can use the **LineSpacingRule** property to set the line spacing to a preset value.

Example

This example sets the line spacing of the paragraph at the cursor position to three lines. This example assumes the insertion point is in a text box.

```
Sub SetLineSpacing()
    Selection.TextRange.ParagraphFormat.LineSpacing = 3
End Sub
```




Show All

LineSpacingRule Property

Returns or sets a **PbLineSpacingRule** that represents the line spacing for the specified paragraphs. Read/write.

PbLineSpacingRule can be one of these PbLineSpacingRule constants.

pbLineSpacing1pt5 Sets paragraph line spacing to a line and a half.

pbLineSpacingDouble Sets paragraph line spacing to two lines.

pbLineSpacingExactly Sets paragraph line spacing to exactly the value of the **LineSpacing** property, even if a larger font is used within the paragraph.

pbLineSpacingMixed A return value for a paragraph that has line spacing of varying values.

pbLineSpacingMultiple A **LineSpacing** property value must be specified, in number of lines.

pbLineSpacingSingle Sets paragraph line spacing to one space.

expression.**LineSpacingRule**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example formats the paragraph at the cursor position to double spacing.

```
Sub SetLineSpacing()
    Selection.TextRange.ParagraphFormat
        .LineSpacingRule = pbLineSpacingDouble
End Sub
```

LinesUp Property

Returns or sets a **Long** that represents the number of lines an initial dropped capital letter is raised above the line of text on which it exists. Read/write.

expression.**LinesUp**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a custom dropped capital letter that is five lines high and raises it two lines above the line on which it exists.

```
Sub RaisedDropCap()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes _
            .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
            Left:=100, Top:=100, Width:=100, Height:=100) _
            .TextFrame.TextRange
        For intCount = 1 To 10
            .InsertAfter NewText:="This is a test. "
        Next intCount
        With .DropCap
            .Size = 5
            .LinesUp = 2
        End With
    End With
End Sub
```


Show All

LinkedFileStatus Property

Returns a **PbLinkedFileStatus** constant that indicates the status of the file linked to the specified picture. Read-only.

PbLinkedFileStatus can be one of these PbLinkedFileStatus constants.

pbLinkedFileMissing The file can no longer be found at the specified path.

pbLinkedFileModified The linked file has been modified since it was linked to the picture.

pbLinkedFileOK The file resides at the specified path, and has not been modified since it was linked to the picture.

expression.**LinkedFileStatus**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked picture files. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

Use either of the following properties to determine whether a shape represents a linked picture:

- The **Type** property of the **Shape** object
- The **IsLinked** property of the **PictureFormat** object

Example

The following example generates a list of the linked pictures in the active publication for which the linked files cannot be found.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat
                If .LinkedFileStatus = pbLinkedFileMissing Then
                Debug.Print .Filename
                End If
            End With

        End If
    Next shpLoop
Next pgLoop
```

LinkFormat Property

Returns a **LinkFormat** object that contains the properties that are unique to linked OLE objects. Read-only.

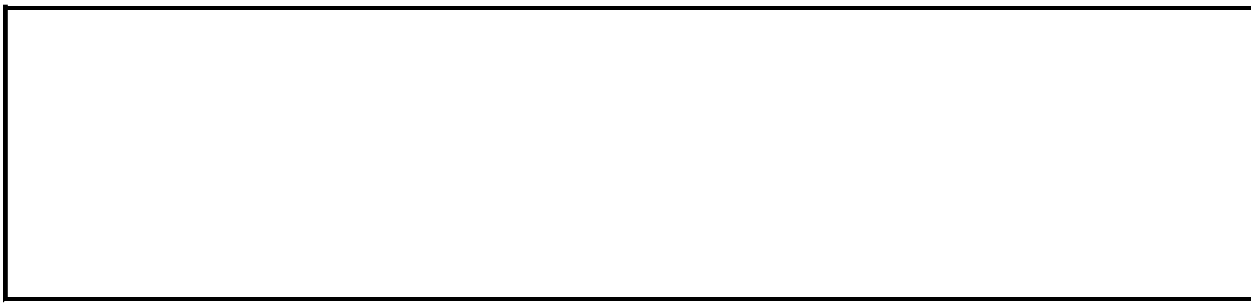
expression.**LinkFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example updates the links between any OLE objects on page one in the active publication and their source files.

```
Dim sh As Shape

For Each sh In ActiveDocument.Pages(1).Shapes
    If sh.Type = pbLinkedOLEObject Then
        With sh.LinkFormat
            .Update
        End With
    End If
Next
```



Links Property

Returns a **WebNavigationBarHyperlinks** collection containing all of the hyperlinks in the specified Web navigation bar set. Read/write.

expression.**Links**

expression Required. An expression that returns a **WebNavigationBarSet** object.

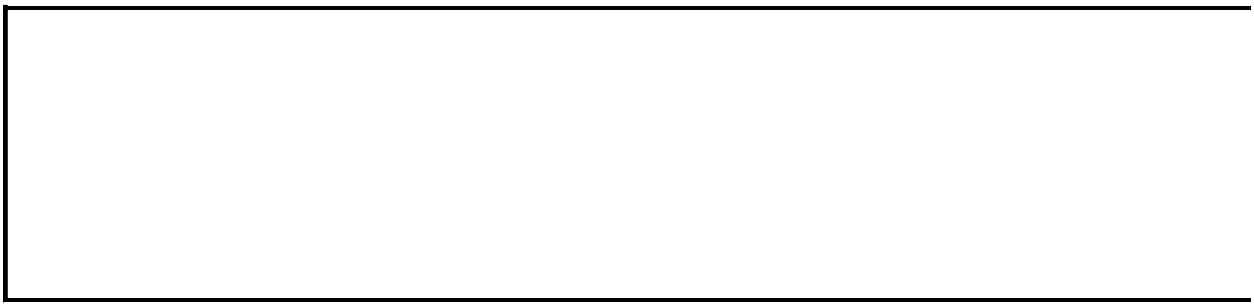
Example

Use the **Links** property to return a **WebNavigationBarHyperlinks** property. This example returns the Web navigation bar hyperlinks of the first Web navigation bar set of the active document.

```
ActiveDocument.WebNavigationBarSets(1).Links
```

The following example adds a new Web navigation bar set to the active document, adds a hyperlink to the navigation bar, and then adds the navigation bar to every page of the publication that has the **AddHyperlinkToWebNavbar** property set to **True** or the **Page.WebPageOptions.IncludePageOnNewWebNavigationBars** property set to **True**.

```
With ActiveDocument.WebNavigationBarSets.AddSet(Name:="WebNavigation
    With .Links
        .Add Address:="www.microsoft.com", TextToDisplay:="Microsoft
    End With
    .AddToEveryPage Left:=10, Top:=10
End With
```



ListBoxItems Property

Returns a **WebListBoxItems** object that represents the items in a Web list box control.

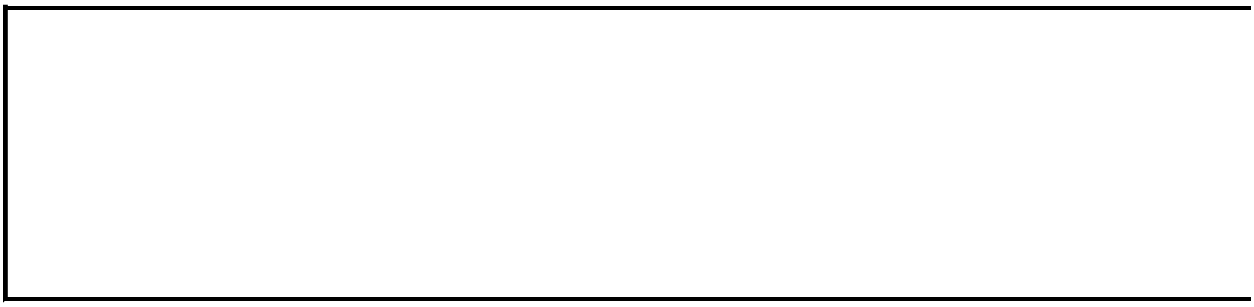
expression.**ListBoxItems**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web list box control and adds five new list items to it.

```
Sub NewListBoxItems()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlListBox, Left:=100, _
            Top:=100, Width:=150, Height:=100).WebListBox
        .MultiSelect = msoTrue
        With .ListBoxItems
            For intCount = 1 To .Count
                .Delete (1)
            Next
            .AddItem Item:="Yellow"
            .AddItem Item:="Red"
            .AddItem Item:="Blue"
            .AddItem Item:="Green"
            .AddItem Item:="Black"
        End With
    End With
End Sub
```



ListBulletFontName Property

Sets or retrieves a **String** representing the list bullet font name from the specified paragraphs. Read/write.

expression.**ListBulletFontName**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

Returns an "Access Denied" message if the list is not a bulleted list.

Example

This example tests to see if the list type is a bulleted list. If it is, the **ListBulletFontName** is set to "Verdana" and the **ListFontSize** is set to 24.

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeBullet Then
        .ListBulletFontName = "Verdana"
        .ListBulletFontSize = 24
    End If
End With
```

ListBulletFontSize Property

Sets or retrieves a **Single** that represents the list bullet font size from the specified paragraphs. Read/write.

expression.**ListBulletFontSize**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

Returns an "Access Denied" message if the list is not a bulleted list.

Example

This example tests to see if the list type is a bulleted list. If it is, the **ListFontSize** is set to 24 and the **ListBulletFontName** is set to "Verdana".

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeBullet Then
        .ListBulletFontSize = 24
        .ListBulletFontName = "Verdana"
    End If
End With
```

ListBulletText Property

Returns a **String** representing the list bullet text from the specified paragraphs. Read-only.

expression.**ListBulletText**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

The **ListBulletText** property is limited to one character.

This property is read-only. To set the **ListBulletText** property of a bulleted list, use the **SetListType** method.

Returns an "Access Denied" message if the list is not a bulleted list.

Example

This example tests to see if the list type is a bulleted list. If it is, a test is made to see that the list bullet text is set to "*". If it is not, the **SetListType** method is called and passed **pbListTypeBullet** as the **pbListType** parameter and "*" as the **BulletText** parameter.

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeBullet Then
        If Not .ListBulletText = "*" Then
            .SetListType pbListTypeBullet, "*"
        End If
    End If
End With
```

ListIndent Property

Returns or sets a **Single** that represents the list indent value (in points) for the specified **ParagraphFormat** object. Read/write.

expression.**ListIndent**

expression Required. An expression that returns a **ParagraphFormat** object.

Example

This example sets the **ListIndent** property of a **ParagraphFormat** object to 0.25 inches. The **InchesToPoints** method is used to convert inches to points.

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    .ListIndent = InchesToPoints(0.25)
End With
```


Show All

ListNumberSeparator Property

Sets or retrieves a **PbListNumberSeparator** constant that represents the list separator of the specified paragraphs. Read/write.

PbListNumberSeparator can be one of these **PbListNumberSeparator** constants.

pbListSeparatorColon
pbListSeparatorDoubleHyphen
pbListSeparatorDoubleParen
pbListSeparatorDoubleSquare
pbListSeparatorParenthesis
pbListSeparatorPeriod
pbListSeparatorPlain
pbListSeparatorSquare
pbListSeparatorWideComma

expression.**ListNumberSeparator**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

The **ListType** property must be set to a numbered list type before setting the **ListNumberSeparator** property. Returns an "Access Denied" message if the list is not a numbered list.

Example

This example tests to see if the list type is a numbered list, specifically **pbListTypeArabic**. If the **ListType** property is set to **pbListTypeArabic** the **ListNumberSeparator** is set to **pbListSeparatorParenthesis**. Otherwise, the **SetListType** method is called and passed **pbListTypeArabic** as the **pbListType** parameter and then the **ListNumberSeparator** property can be set.

```
Dim objParaForm As ParagraphFormat

Set objParaForm  = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeArabic Then
        .ListNumberSeparator = pbListSeparatorParenthesis
    Else
        .SetListType pbListTypeArabic
        .ListNumberSeparator = pbListSeparatorParenthesis
    End If
End With
```

ListNumberStart Property

Sets or retrieves a **Long** that represents the starting number of a list. Read/write.

expression.**ListNumberStart**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

Returns an "Access Denied" message if the list is not a numbered list.

Example

This example sets the list type of a **ParagraphFormat** object to **pbListTypeArabic** and sets the **ListNumber** property to 4.

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

    With objParaForm
        .SetListType pbListTypeArabic
        .ListNumberStart = 4
    End With

End Sub
```


Show All

ListType Property

Returns a **pbListType** constant from the specified **ParagraphFormat** object. Read-only.

PbListType can be one of these PbListType constants.
pbListTypeAiueo
pbListTypeArabic
pbListTypeArabic1
pbListTypeArabic2
pbListTypeArabicLeadingZero
pbListTypeBullet
pbListTypeCardinalText
pbListTypeChiManSty
pbListTypeChinaDbNum1
pbListTypeChinaDbNum2
pbListTypeChinaDbNum3
pbListTypeChinaDbNum4
pbListTypeChosung
pbListTypeCirclenum
pbListTypeDAiueo
pbListTypeDArabic
pbListTypeDbChar
pbListTypeDbNum1
pbListTypeDbNum2
pbListTypeDbNum3
pbListTypeDbNum4
pbListTypeDIroha
pbListTypeGanada
pbListTypeGB1
pbListTypeGB2
pbListTypeGB3

pbListTypeGB4
pbListTypeHebrew1
pbListTypeHebrew2
pbListTypeHex
pbListTypeHindi1
pbListTypeHindi2
pbListTypeHindi3
pbListTypeHindi4
pbListTypeIroha
pbListTypeKoreaDbNum1
pbListTypeKoreaDbNum2
pbListTypeKoreaDbNum3
pbListTypeKoreaDbNum4
pbListTypeLowerCaseLetter
pbListTypeLowerCaseRoman
pbListTypeLowerCaseRussian
pbListTypeNone
pbListTypeOrdinal
pbListTypeOrdinalText
pbListTypeSbChar
pbListTypeTaiwanDbNum1
pbListTypeTaiwanDbNum2
pbListTypeTaiwanDbNum3
pbListTypeTaiwanDbNum4
pbListTypeThai1
pbListTypeThai2
pbListTypeThai3
pbListTypeUpperCaseLetter
pbListTypeUpperCaseRoman
pbListTypeUpperCaseRussian
pbListTypeVietnamese1
pbListTypeZodiac1
pbListTypeZodiac2

pbListTypeZodiac3

expression.**ListType**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

This property is read-only. To set the **ListType** property of a **ParagraphFormat** object, use the **SetListType** method.

Example

This example tests to see if the list type is a numbered list, specifically **pbListTypeArabic**. If the **ListType** property is set to **pbListTypeArabic**, the **ListNumberSeparator** is set to **pbListSeparatorParenthesis**.

```
Dim objParaForm As ParagraphFormat

Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
.TextFrame.TextRange.ParagraphFormat

With objParaForm
    If .ListType = pbListTypeArabic Then
        .ListNumberSeparator = pbListSeparatorParenthesis
    End If
End With
```


Show All

LockAspectRatio Property

Returns or sets an **MsoTriState** constant indicating whether the specified shape retains its original proportions when you resize it. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The height and width of the shape change independently of one another when you resize it.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue The specified shape retains its original proportions when you resize it.

expression.**LockAspectRatio**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a cube to the active publication. The cube can be moved and resized, but not reproportioned.

```
Dim shp As Shape

Set shp = ActiveDocument.Pages(1).Shapes _
    .AddShape(Type:=msoShapeCube, _
    Left:=50, Top:=50, Width:=100, Height:=200) _

shp.LockAspectRatio = msoTrue
```


Show All

LockToBaseLine Property

Returns an **MsoTristate** that represents whether or not text will be positioned along baseline guides. Read/write.

msoCTrue

msoFalse The text is not aligned to baselines.

msoTriStateMixed The specified paragraphs contain both text that is aligned to baselines and text that is not aligned to baselines.

msoTriStateToggle

msoTrue The text is aligned to baselines.

expression.**LockToBaseLine**

expression Required. An expression that returns a **ParagraphFormat** object.

Example

The following example sets the **LockToBaseLine** property to **True**.

```
Dim objParaForm As ParagraphFormat
Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.ParagraphFormat
objParaForm.LockToBaseLine = msoTrue
```

Luminance Property

Returns or sets a **Long** indicating a calculated luminance value for the specified plate; used for spot-color trapping. Valid values are from 0 to 100. Read/write.

expression.**Luminance**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is valid only for publications with a **ColorMode** property value of **pbColorModeSpot** or for spot plates in a publication with a **ColorMode** property value of **pbColorModeSpotAndProcess**.

Example

The following example loops through all the spot-color plates in a publication and reports their luminance values.

```
Dim plaTemp As Plates
Dim plaLoop As Plate

Set plaTemp = ActiveDocument.Plates

If ActiveDocument.ColorMode <> pbColorModeSpot And _
        ActiveDocument.ColorMode <> pbColorModeSpotAndProcess Then
    Debug.Print "No spot colors in this publication."
Else
    For Each plaLoop In plaTemp
        With plaLoop
            Debug.Print "Plate " & .Name _
                & " has a luminance of " & .Luminance
        End With
    Next plaLoop
End If
```


Show All

Magenta Property

Sets or returns a **Long** that represents the magenta component of a CMYK color. Value can be any number between 0 and 255. Read/write.

expression.**Magenta**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two new shapes and then sets the CMYK fill color for one shape and sets the CMYK values of the second shape to the same CMYK values.

```
Sub ReturnAndSetCMYK()
    Dim lngCyan As Long
    Dim lngMagenta As Long
    Dim lngYellow As Long
    Dim lngBlack As Long
    Dim shpHeart As Shape
    Dim shpStar As Shape

    Set shpHeart = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeHeart, Left:=100, _
        Top:=100, Width:=100, Height:=100)
    Set shpStar = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=100, Width:=150, Height:=150)

    With shpHeart.Fill.ForeColor.CMYK
       .SetCMYK Cyan:=10, Magenta:=80, Yellow:=200, Black:=30
        lngCyan = .Cyan
        lngMagenta = .Magenta
        lngYellow = .Yellow
        lngBlack = .Black
    End With

    'Sets new shape to current shapes CMYK colors
    shpStar.Fill.ForeColor.CMYK.SetCMYK _
        Cyan:=lngCyan, Magenta:=lngMagenta, _
        Yellow:=lngYellow, Black:=lngBlack
End Sub
```

MailEnvelope Property

Returns an **MsoEnvelope** object that represents an e-mail header for a publication.

expression.**MailEnvelope**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **MailEnvelope** property is only accessible if the **EnvelopeVisible** property has been set to **True**.

Example

This example sets the comments for the e-mail header of the active publication. This example assumes that the **EnvelopeVisible** property has been set to **True**.

```
Sub HeaderComments()
    ActiveDocument.MailEnvelope.Introduction = _
        "Please review this publication and let me know " & _
        "what you think.  I need your input by Friday." & _
        "  Thanks."
End Sub
```

MailMerge Property

Returns a **MailMerge** object that represents the mail merge functionality for the specified publication.

expression.**MailMerge**

expression Required. An expression that returns one of the objects in the Applies To list.

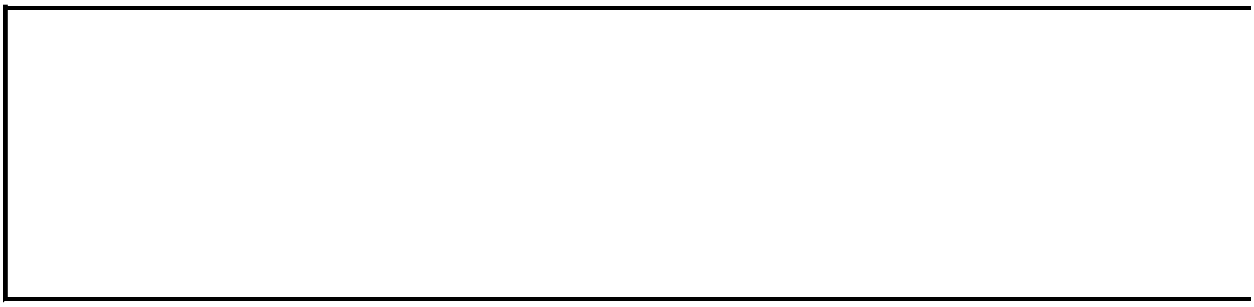
Example

This example displays the information from the current data record in the data source.

```
Sub ViewMergeData()
    ActiveDocument.MailMerge.ViewMailMergeFieldCodes = False
End Sub
```

This example displays the **Mail Merge Recipients** dialog box, which contains the records from the data source.

```
Sub ExecuteMergeField()
    ActiveDocument.MailMerge.DataSource.OpenRecipientsDialog
End Sub
```



MajorityFont Property

Returns a **Font** object that represents the font name most in use in a text range.

expression.**MajorityFont**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new text box, fills it with text, checks if the font most in use is Tahoma, and if it isn't, changes the font to Tahoma.

```
Sub SetFontName()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes _
            .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
            Left:=100, Top:=100, Width:=100, Height:=100) _
            .TextFrame.TextRange
        For intCount = 1 To 10
            .InsertAfter NewText:="This is a test. "
        Next intCount
        If .MajorityFont <> "Tahoma" Then _
            .Font.Name = "Tahoma"
    End With
End Sub
```

MajorityParagraphFormat Property

Returns a **ParagraphFormat** object that represents the paragraph formatting applied to most of the paragraphs in a text range.

expression.**MajorityParagraphFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example applies the paragraph formatting applied to a majority of the paragraphs in the first shape to the paragraphs in the second shape on the first page of the active document. This example assumes that there are at least two shapes on page one of the active publication.

```
Sub SetFontName()
    Dim fmt As ParagraphFormat
    With ActiveDocument.Pages(1)
        Set fmt = .Shapes(1).TextFrame.TextRange _
                .MajorityParagraphFormat
        .Shapes(2).TextFrame.TextRange.ParagraphFormat = fmt
    End With
End Sub
```


Show All

MappedDataFields Property

Returns a **MailMergeMappedDataFields** object that represents the mapped data fields available in Publisher.

expression.**MappedDataFields**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example creates a table on a new page of the current publication and lists the mapped data fields available in Publisher and the fields in the data source to which they are mapped. This example assumes that the current publication is a mail merge publication and that the data source fields have corresponding mapped data fields.

```
Sub MappedFields()
    Dim intCount As Integer
    Dim intRows As Integer
    Dim docPub As Document
    Dim pagNew As Page
    Dim shpTable As Shape
    Dim tblTable As Table
    Dim rowTable As Row

    On Error Resume Next

    Set docPub = ThisDocument
    Set pagNew = ThisDocument.Pages.Add(Count:=1, After:=1)
    intRows = docPub.MailMerge.DataSource.MappedDataFields.Count + 1

    'Creates new table with a heading row
    Set shpTable = pagNew.Shapes.AddTable(NumRows:=intRows, _
        numColumns:=2, Left:=100, Top:=100, Width:=400, Height:=12)
    Set tblTable = shpTable.Table
    With tblTable.Rows(1)
        With .Cells(1).Text
            .Text = "Mapped Data Field"
            .Font.Bold = msoTrue
        End With
        With .Cells(2).Text
            .Text = "Data Source Field"
            .Font.Bold = msoTrue
        End With
    End With

    With docPub.MailMerge.DataSource
        For intCount = 2 To intRows - 1
            'Inserts mapped data field name and the
            'corresponding data source field name
            tblTable.Rows(intCount - 1).Cells(1).Text _
                .Text = .MappedDataFields(Index:=intCount).Name
            tblTable.Rows(intCount - 1).Cells(2).Text _
                .Text = .MappedDataFields(Index:=intCount).DataField
```

```
        Next
    End With
End Sub
```

MarginBottom Property

Returns or sets a **Variant** that represents the amount of space (in points) between the text and the bottom edge of a cell, text frame, or page. Read/write.

expression.**MarginBottom**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the margins of the active publication to two inches.

```
Sub SetPageMargins()

    With ActiveDocument.LayoutGuides
        .MarginTop = Application.InchesToPoints(Value:=2)
        .MarginBottom = Application.InchesToPoints(Value:=2)
        .MarginLeft = Application.InchesToPoints(Value:=2)
        .MarginRight = Application.InchesToPoints(Value:=2)
    End With

End Sub
```

MarginLeft Property

Returns or sets a **Variant** that represents the amount of space (in points) between the text and the left edge of a cell, text frame, or page. Read/write.

expression.**MarginLeft**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the margins of the active publication to two inches.

```
Sub SetPageMargins()

    With ActiveDocument.LayoutGuides
        .MarginTop = Application.InchesToPoints(Value:=2)
        .MarginBottom = Application.InchesToPoints(Value:=2)
        .MarginLeft = Application.InchesToPoints(Value:=2)
        .MarginRight = Application.InchesToPoints(Value:=2)
    End With

End Sub
```

MarginRight Property

Returns or sets a **Variant** that represents the amount of space (in points) between the text and the right edge of a cell, text frame, or page. Read/write.

expression.**MarginRight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the margins of the active publication to two inches.

```
Sub SetPageMargins()

    With ActiveDocument.LayoutGuides
        .MarginTop = Application.InchesToPoints(Value:=2)
        .MarginBottom = Application.InchesToPoints(Value:=2)
        .MarginLeft = Application.InchesToPoints(Value:=2)
        .MarginRight = Application.InchesToPoints(Value:=2)
    End With

End Sub
```

MarginTop Property

Returns or sets a **Variant** that represents the amount of space (in points) between the text and the top edge of a cell, text frame, or page. Read/write.

expression.**MarginTop**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the margins of the active publication to two inches.

```
Sub SetPageMargins()

    With ActiveDocument.LayoutGuides
        .MarginTop = Application.InchesToPoints(Value:=2)
        .MarginBottom = Application.InchesToPoints(Value:=2)
        .MarginLeft = Application.InchesToPoints(Value:=2)
        .MarginRight = Application.InchesToPoints(Value:=2)
    End With

End Sub
```

Master Property

Sets or returns a **Page** object that represents the master page properties for the specified page.

expression.**Master**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Master pages do not have a **Master** property. Any attempt to access the **Master** property of a master page will result in a run-time error.

Example

This example adds a shape to the master page for the first page in the active publication.

```
Sub AddNewMasterPageShape()
    With ActiveDocument.Pages(1).Master.Shapes.AddShape _
            (Type:=msoShape5pointStar, Left:=512, _
            Top:=50, Width:=50, Height:=50)
        .Fill.ForeColor.CMYK.SetCMYK Cyan:=255, _
            Magenta:=255, Yellow:=0, Black:=0
    End With
End Sub
```

The **Master** property can also be used to apply a master page to a page in a publication. The following example sets the master page of the first page of a publication to the master page of the second page in the publication. This example assumes that there are at least two pages and two master pages in the document.

```
ActiveDocument.Pages(1).Master = _
    ActiveDocument.Pages(2).Master
```

MasterPages Property

Returns the **MasterPages** collection for the specified publication.

expression.**MasterPages**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets the text in the first text frame on the master page to Second Quarter.

```
Dim mp As MasterPages

Set mp = ActiveDocument.MasterPages

With mp.Item(1)
    .Shapes(1).TextFrame.TextRange.Text = "Second Quarter"
End With
```

MatchAlefHamza Property

Sets or returns a **Boolean** representing whether or not a search operation will match alefs and hamzas. Read/write.

expression.**MatchAlefHamza**

expression Required. An expression that returns a **FindReplace** object.

Remarks

This property may not be available depending on the language enabled on your operating system. The default value is **False**.

Returns **Access denied** if Arabic is not enabled.

Examples

This example finds the first occurrence of the word "إخرجنا" in an Arabic document matching alefs and hamzas.

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "إخرجنا"
    .MatchAlefHamza = True
    .Execute
End With
```

This example follows from the previous one except that alef hamzas will not be matched. Therefore the words "إخرجنا" or "خرجن" will both be found because alefs and hamzas will be ignored.

"إخرجنا".

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "إخرجنا"
    .MatchAlefHamza = False
    .Execute
End With
```

MatchCase Property

Sets or returns a **Boolean** that represents the case sensitivity of the search operation. Read/write.

expression.**MatchCase**

expression Required. An expression that returns a **FindReplace** object.

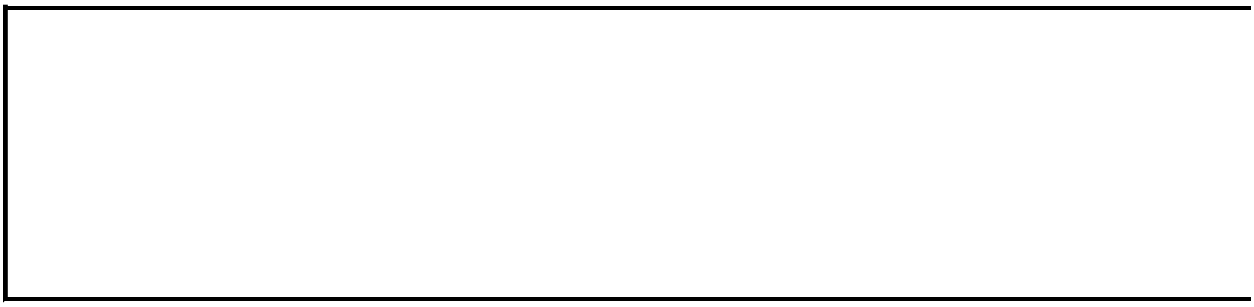
Remarks

The default value for **MatchCase** is **False**.

Example

This example will select the first occurrence of the word "factory" regardless of case.

```
With ActiveDocument.Find
    .Clear
    .MatchCase = False
    .FindText = "factory"
    .Execute
End With
```



MatchDiacritics Property

Sets or returns a **Boolean** representing whether or not a search operation will match diacritics. Read/write.

expression.**MatchDiacritics**

expression Required. An expression that returns a **FindReplace** object.

Remarks

This property may not be available depending on the languages enabled on your operating system. The default value is **False**.

Returns **Access denied** if a proper language, such as Arabic, is not enabled.

Example

This example finds the first occurrence of the word "gegenüber" in a German document.

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "gegenüber"
    .MatchDiacritics = True
    .Execute
End With
```

MatchKashida Property

Sets or returns a **Boolean** representing whether or not a search operation will match kashidas. Read/write.

expression.**MatchKashida**

expression Required. An expression that returns a **FindReplace** object.

Remarks

This property may not be available depending on the language enabled on your operating system. The default value is **False**.

Returns **Access denied** if Arabic is not enabled.

Examples

This example finds the first occurrence of the word "مـــحـــمـــد" in an Arabic document matching kashidas.

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "مـــــحـــــمـــــد"
    .MatchKashida = True
    .Execute
End With
```

This example follows from the previous one except that kashidas will not be matched. Therefore the words "محمد" or "مـــحـــمـــد" will both be found because kashidas will be ignored.

```
Dim objDocument As Document

Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "مـــــحـــــمـــــد"
    .MatchKashida = False
    .Execute
End With
```

MatchWholeWord Property

Sets or returns a **Boolean** that represents whether the whole word will be matched in the search operation. Read/write. **Boolean**.

expression.**MatchWholeWord**

expression Required. An expression that returns a **FindReplace** object.

Remarks

The default value for **MatchWholeWord** is **False**.

Example

This example will select each occurrence of the word "fact" and apply bold formatting.

```
With ActiveDocument.Find
    .Clear
    .MatchWholeWord = True
    .FindText = "fact"
    .ReplaceScope = pbReplaceScopeNone
    Do While .Execute = True
        .FoundTextRange.Font.Bold = msoTrue
    Loop
End With
```

This example follows the previous example except that whole words will not be matched. Therefore the word "fact" within the word "factory" or "factoid" will also have bold formatting applied.

```
With ActiveDocument.Find
    .Clear
    .MatchWholeWord = False
    .FindText = "fact"
    .ReplaceScope = pbReplaceScopeNone
    Do While .Execute = True
        .FoundTextRange.Font.Bold = msoTrue
    Loop
End With
```

MatchWidth Property

Sets or returns a **Boolean** representing whether or not a search operation will match the character width of the searched text. Read/Write.

expression.**MatchWidth**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

This property may not be available depending on the language enabled on your operating system. The default value is **False**.

Return "Access denied" if an East Asian language is not enabled.

Example

The following example finds each occurance of the word "width" in the active document and applies bold formatting. The **MatchWidth** property is set to **False** so that full or half width characters will both be found. For example, this search will apply bold formatting to the word "width" (half-width characters) and the word " " (full-width characters).

```
Dim objDocument As Document
Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "width"
    .MatchWidth = False
    Do While .Execute = True
        .FoundTextRange.Font.Bold = msoTrue
    Loop
End With
```

The following example finds each occurance of the word "width" in the active document and applies bold formatting. The **MatchWidth** property is set to **True** so that either full or half width characters will be found. For example, this search will apply bold formatting to "width". It will not apply formatting to the word " ".

```
Dim objDocument As Document
Set objDocument = ActiveDocument
With objDocument.Find
    .Clear
    .FindText = "width"
    .MatchWidth = True
    Do While .Execute = True
        .FoundTextRange.Font.Bold = msoTrue
    Loop
End With
```


Show All

MeasurementUnit Property

Returns or sets a **PbUnitType** constant representing the standard measurement unit for Microsoft Publisher. Read/write.

PbUnitType can be one of these PbUnitType constants.

pbUnitCM Sets the unit of measurement to centimeters.

pbUnitEmu Doesn't apply to this property; returns an error if used.

pbUnitFeet Doesn't apply to this property; returns an error if used.

pbUnitHa Doesn't apply to this property; returns an error if used.

pbUnitInch Sets the unit of measurement to inches.

pbUnitKyu Doesn't apply to this property; returns an error if used.

pbUnitMeter Doesn't apply to this property; returns an error if used.

pbUnitPica Sets the unit of measurement to picas.

pbUnitPoint Sets the unit of measurement to points.

pbUnitTwip Doesn't apply to this property; returns an error if used.

expression.**MeasurementUnit**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the standard measurement unit for Publisher to points.

```
Sub SetUnitOfMeasurement()
    Options.MeasurementUnit = pbUnitPoint
End Sub
```

This example displays the current unit of measurement.

```
Sub GetUnitOfMeasurement()
    Dim measUnit As PbUnitType
    Dim strUnit As String

    measUnit = Options.MeasurementUnit

    Select Case measUnit
        Case 0
            strUnit = "inches"
        Case 1
            strUnit = "centimeters"
        Case 2
            strUnit = "picas"
        Case 3
            strUnit = "points"
    End Select

    MsgBox "The current unit of measurement is " & strUnit

End Sub
```

MirrorGuides Property

Returns or sets a **Boolean** indicating whether Microsoft Publisher creates mirror guide positions for a book fold publication. **True** if Publisher creates mirror guide positions for separate left and right pages in a book fold publication; **False** if the same margin, row, and column guides are applied to all pages in the publication. Read/write.

expression.**MirrorGuides**

expression Required. An expression that returns one of the objects in the Applies To list.

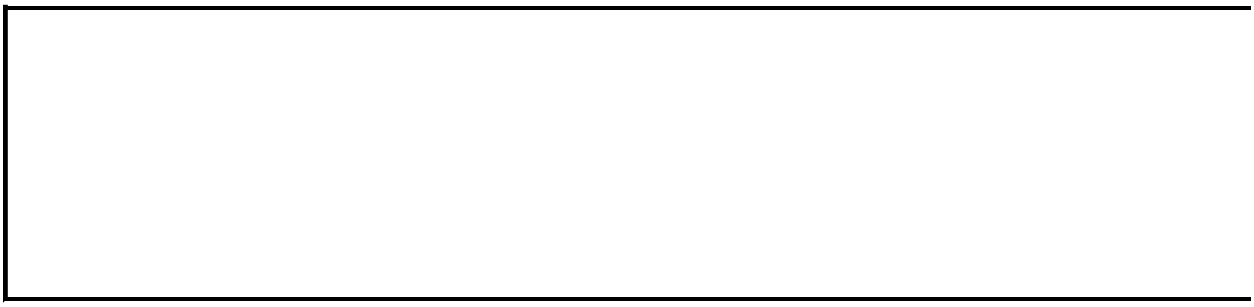
Remarks

When the **MirrorGuides** property is **True**, margin settings apply to right-facing pages and are mirrored for left-facing pages. In addition, when set to **True**, the **MirrorGuides** property sets the publication to use two master pages instead of the default of one. The first master page is for all left-facing pages and the second is for all right-facing pages in the publication. For more information, see **MasterPages** object.

Example

The following example sets Publisher to create mirror guides for a book fold publication and sets the inside and outside margins of each two-page spread. The example sets the left and right margin values for right-facing pages, and Publisher mirrors these values for left-facing pages.

```
With ActiveDocument.LayoutGuides
    .MirrorGuides = True
    .MarginLeft = 48
    .MarginRight = 96
End With
```



MultiplePagesPerSheet Property

True for Microsoft Publisher to print multiple pages onto a single sheet. Read/write **Boolean**.

expression.**MultiplePagesPerSheet**

expression Required. An expression that returns one of the objects in the Applies To list.

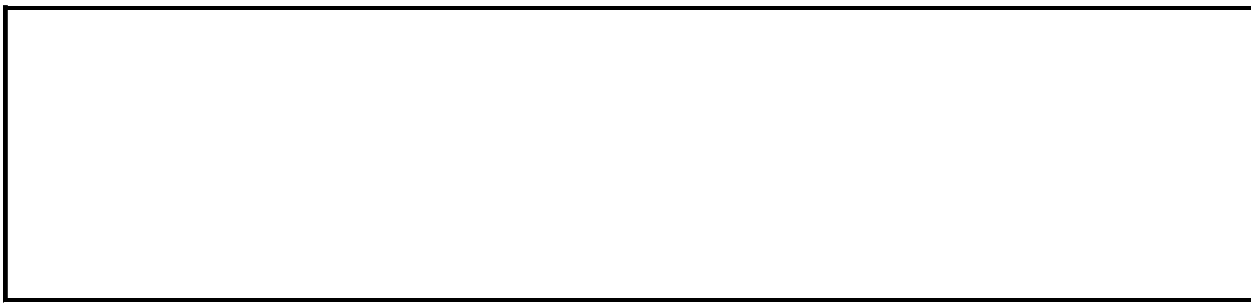
Remarks

If the page size is greater than half the paper size, Publisher will display an error.

Example

This example sets the active publication to print multiple pages on a single sheet when printing the publication.

```
Sub SetLeftMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```




Show All

MultiSelect Property

MsoTrue if a user may select more than one item in a Web list box control. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse Indicates a user may only select one item in a Web list box control.

msoTriStateMixed

msoTriStateToggle

msoTrue Indicates a user may select more than one item in a Web list box control.

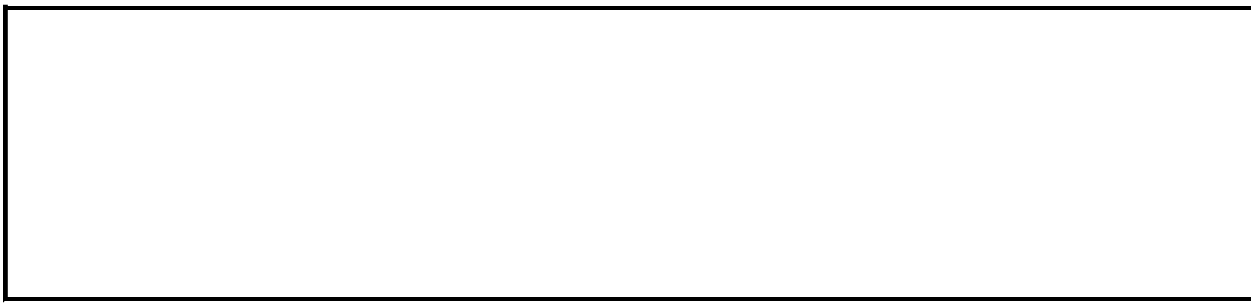
expression.**MultiSelect**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example add a Web list box control to the active publication, add items to it, and specifies that a user may select more than one item.

```
Sub NewListBoxItems()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlListBox, Left:=100, _
            Top:=100, Width:=150, Height:=100).WebListBox
        .MultiSelect = msoTrue
        With .ListBoxItems
            For intCount = 1 To .Count
                .Delete (1)
            Next
            .AddItem Item:="Yellow"
            .AddItem Item:="Red"
            .AddItem Item:="Blue"
            .AddItem Item:="Green"
            .AddItem Item:="Black"
        End With
    End With
End Sub
```




Show All

Name Property

 Name property as it applies to the **Application**, **BorderArt**, **ColorScheme**, **Document**, **Label**, **MailMergeDataSource**, **MailMergeMappedDataField**, **Plate**, **Tag**, **TextStyle**, **Wizard**, **WizardProperty**, and **WizardValue** objects.

Returns a **String** value indicating the name of the specified object. Read-only.

expression.**Name**

expression Required. An expression that returns one of the above objects.

 Name property as it applies to the **BorderArtFormat**, **Font**, **Page**, **Shape**, and **ShapeRange** objects.

Returns or sets a **String** value indicating the name of the specified object. Read/write.

expression.**Name**

expression Required. An expression that returns one of the above objects.

Remarks

You can use an object's name in conjunction with the **Item** method or **Item** property to return a reference to the object if the **Item** method or property for the collection that contains the object takes a **Variant** argument. For example, if the value of the **Name** property for a shape is Rectangle 2, then `.Shapes("Rectangle 2")` will return a reference to that shape.

The **Name** property is the default property for the **BorderArt**, **BorderArtFormat**, and **Label** objects.

Example

 As it applies to the **ColorScheme** object.

This example reports the name of the color scheme for the active publication.

```
MsgBox "The current color scheme is " _
    & ActiveDocument.ColorScheme.Name & "."
```

 As it applies to the **Font** object.

This example formats a text frame on page one as Arial bold.

```
With ActiveDocument.Pages(1).Shapes(1) _
        .TextFrame.TextRange.Font
    .Name = "Arial"
    .Bold = True
End With
```

NegativeImage Property

True to print a negative image of the specified publication. The default is **False**. Read/write **Boolean**.

expression.**NegativeImage**

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property is only accessible if the active printer is a PostScript printer. Returns a run-time error if a non-PostScript printer is specified. Use the **IsPostscriptPrinter** property of the **AdvancedPrintOptions** object to determine if the specified printer is a PostScript printer.

This property is saved as an application setting and applied to future instances of Publisher.

This property corresponds to the **Negative image** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

This property is mostly used when printing to film on an imagesetter, so that the image reads positive when burned onto a press plate.

Example

The following example determines if the active printer is a PostScript printer. If it is, the active publication is set to print as a horizontally and vertically mirrored, negative image of itself.

```
Sub PrepToPrintToFilmOnImagesetter()

With ActiveDocument.AdvancedPrintOptions
    If .IsPostscriptPrinter = True Then
        .HorizontalFlip = True
        .VerticalFlip = True
        .NegativeImage = True
    End If
End With

End Sub
```

Next Property

Returns a **Field** object that represents the next field in a text range.

expression.**Next**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example bolds the field next to the first field in the specified text range. This assumes that there are at least two fields in the specified text range.

```
Sub GoToNextField()
    ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
        .Fields(1).Next.TextRange.Font.Bold = msoTrue
End Sub
```

NextLinkedTextFrame Property

Returns or sets a **TextFrame** object representing the text frame to which text flows from the specified text frame. Read/write.

expression.**NextLinkedTextFrame**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the specified text frame is not part of a chain of linked frames or is the last in a chain of linked frames, this property returns nothing.

Example

The following example returns the next linked text frame of shape three on page one of the active publication and sets its font to Times New Roman.

```
Dim txtFrame As TextFrame

Set txtFrame = ActiveDocument.Pages(1) _
    .Shapes(3).TextFrame.NextLinkedTextFrame

txtFrame.TextRange.Font = "Times New Roman"
```

NextParagraphStyle Property

Returns or sets a **String** that represents the paragraph style that follows the specified text style when a user presses ENTER. Read/write.

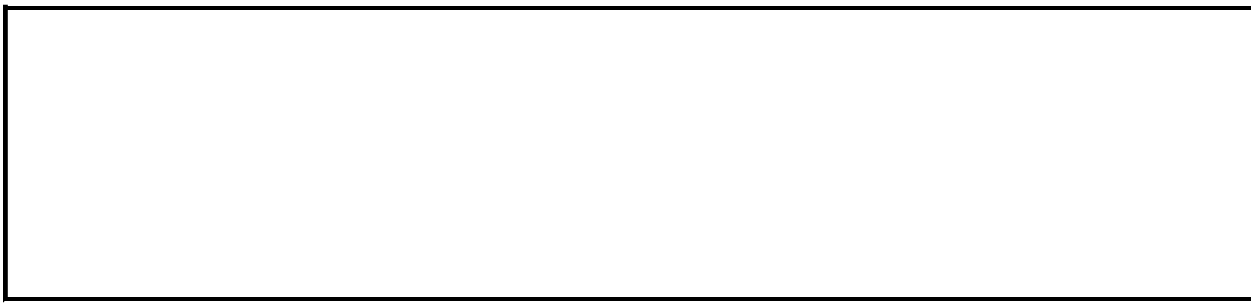
expression.**NextParagraphStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new text style and specifies that the text style following the new text style is the Normal style.

```
Sub CreateNewTextStyle()
    Dim styNew As TextStyle
    Dim fntStyle As Font

    Set styNew = ActiveDocument.TextStyles.Add(StyleName:="Heading 1
    Set fntStyle = styNew.Font

    With fntStyle
        .Name = "Tahoma"
        .Bold = msoTrue
        .Size = 15
    End With

    With styNew
        .Font = fntStyle
        .NextParagraphStyle = "Normal"
    End With
End Sub
```



Nodes Property

Returns a **ShapeNodes** collection that represents the geometric description of the specified shape. Applies to **Shape** or **ShapeRange** objects that represent freeform drawings.

expression.Nodes

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a smooth node with a curved segment after node four in shape three on page one. Shape three must be a freeform drawing with at least four nodes.

```
With ActiveDocument.Pages(1) _
        .Shapes(3).Nodes
    .Insert Index:=4, SegmentType:=msoSegmentCurve, _
        EditingType:=msoEditingSmooth, X1:=210, Y1:=100
End With
```


Show All

NormalizedHeight Property

Returns or sets **MsoTrue** if all characters (both uppercase and lowercase) in the specified WordArt are the same height. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse Characters in the specified WordArt object are not all the same height.

msoTriStateMixed Not used with this property.

msoTriStateToggle Not used with this property.

msoTrue Characters in the specified WordArt object are all the same height.

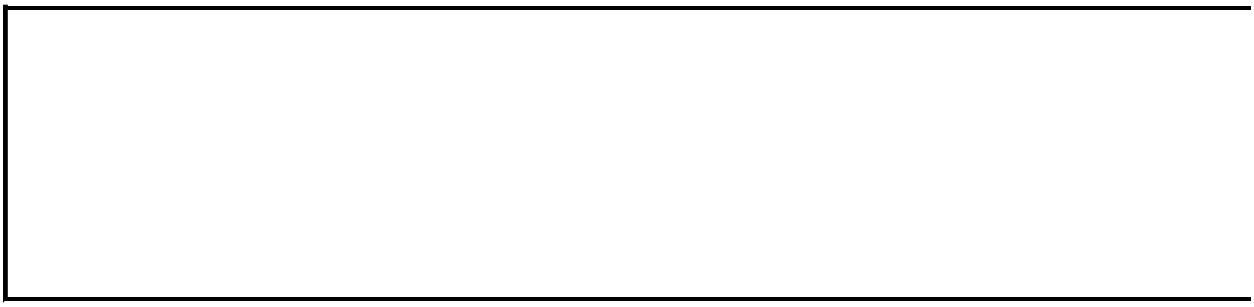
expression.**NormalizedHeight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new WordArt shape on the first page of the active publication and then sets each character in the shape to be the same height.

```
Sub SetNormalHeight()
    With ActiveDocument.Pages(1).Shapes.AddTextEffect _
            (PresetTextEffect:=msoTextEffect10, _
            text:="Test WordArt Shape", FontName:="Snap ITC", _
            FontSize:=30, FontBold:=msoFalse, FontItalic:=msoFalse, _
            Left:=36, Top:=36).TextEffect
        .NormalizedHeight = msoTrue
    End With
End Sub
```



Object Property

Returns an **Object** that represents the specified OLE object's top-level interface. This property allows you to access the properties and methods of an ActiveX control or the application in which an OLE object was created. The OLE object must support OLE Automation for this property to work. Read-only.

expression.**Object**

expression Required. An expression that returns an **OLEFormat** object.

Example

This example sets the value of the first shape in the active publication. For the example to work, this first shape must be an ActiveX control (for example, a check box or an option button).

```
Dim myObj As Object

With ActiveDocument.Pages(1).Shapes(1).OLEFormat
    .Activate
    Set myObj = .Object
End With

myObj.Value = True
```


Show All

ObjectVerbs Property

Returns an **ObjectVerbs** collection that contains all the OLE verbs for the specified OLE object. Read-only.

expression.ObjectVerbs

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays all the available verbs for the OLE object contained in shape one on page two in the active publication. For this example to work, shape one must be a shape that represents an OLE object.

```
Dim v As String

With ActiveDocument.Pages(2).Shapes(1).OLEFormat
    For Each v In .ObjectVerbs
        MsgBox v
    Next
End With
```


Show All

Obscured Property

Returns or sets an **MsoTriState** value indicating whether the shadow of the specified shape appears filled in and is obscured by the shape. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shadow of the specified shape does not appear filled in and is not obscured by the shape if the shape has no fill.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue The shadow of the specified shape appears filled in and is obscured by the shape.

expression.Obscured

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the horizontal and vertical offsets of the shadow for shape three on page one of the active publication. The shadow is offset 5 points to the right of the shape and 3 points above it. If the shape doesn't already have a shadow, this example adds one to it. The shadow will be filled in and obscured by the shape, even if the shape has no fill.

```
With ActiveDocument.Pages(1).Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
    .Obscured = msoTrue
End With
```


Show All

OfficeDataSourceObject Property

Returns an **OfficeDataSourceObject** object representing the data source in a mail merge or catalog merge operation. Read-only.

expression.**OfficeDataSourceObject**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays information about the current mail merge data source.

```
Dim odsoTemp As Office.OfficeDataSourceObject

Set odsoTemp = Application.OfficeDataSourceObject

With odsoTemp
    Debug.Print "Connection string: " & .ConnectString
    Debug.Print "Data source: " & .DataSource
    Debug.Print "Table: " & .Table
End With
```

OffsetX Property

Returns or sets a **Variant** value indicating the vertical offset of the shadow from the specified shape. A positive value offsets the shadow below the shape; a negative value offsets it above the shape. Read/write.

expression.**OffsetX**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

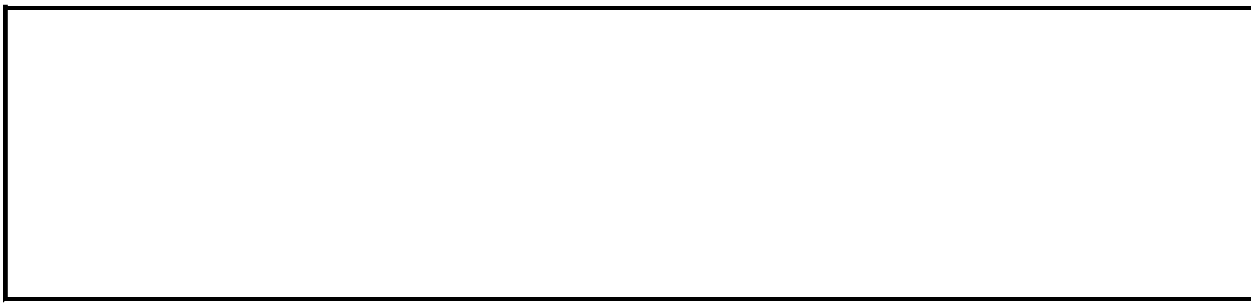
Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

If you want to nudge a shadow horizontally or vertically from its current position without having to specify an absolute position, use the **IncrementOffsetX** method or the **IncrementOffsetY** method.

Example

This example sets the horizontal and vertical offsets of the shadow for shape three on page one of the active publication. The shadow is offset 5 points to the right of the shape and 3 points above it. If the shape doesn't already have a shadow, this example adds one to it.

```
With ActiveDocument.Pages(1).Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```



OffsetY Property

Returns or sets a **Variant** value indicating the horizontal offset of the shadow from the specified shape. A positive value offsets the shadow to the right of the shape; a negative value offsets it to the left. Read/write.

expression.**OffsetY**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

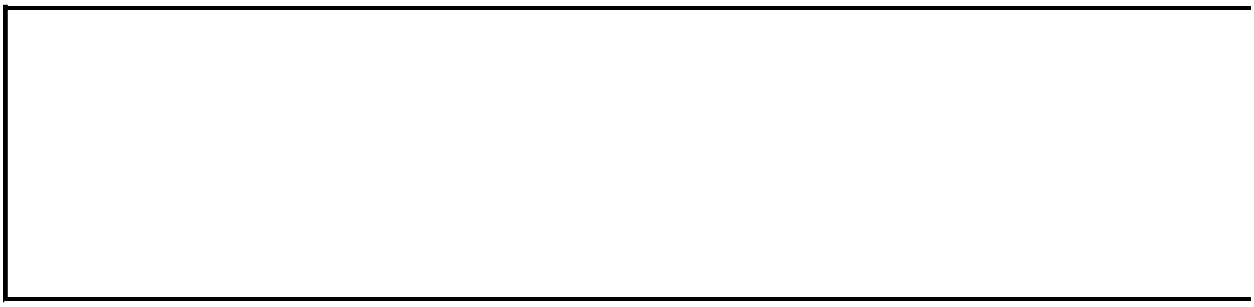
Numeric values are evaluated in points; strings can be in any units supported by Microsoft Publisher (for example, "2.5 in").

If you want to nudge a shadow horizontally or vertically from its current position without having to specify an absolute position, use the **IncrementOffsetX** method or the **IncrementOffsetY** method.

Example

This example sets the horizontal and vertical offsets of the shadow for shape three on page one of the active publication. The shadow is offset 5 points to the right of the shape and 3 points above it. If the shape doesn't already have a shadow, this example adds one to it.

```
With ActiveDocument.Pages(1).Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```



OLEFormat Property

Returns an **OLEFormat** object that contains OLE formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent OLE objects.

expression.**OLEFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example loops through all the shapes on the first page of the active document and automatically updates all linked Excel worksheets.

```
Sub UpdateLinkedExcelSpreadsheets()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        If shp.Type = msoLinkedOLEObject Then
            If shp.OLEFormat.ProgId = "Excel.Sheet" Then
                shp.LinkFormat.Update
            End If
        End If
    Next shp
End Sub
```

Options Property

Returns an **Options** object that represents application settings you can set in Publisher.

expression.**Options**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example disables background saves and then saves the active publication.

```
Sub SetGlobalSaveOptions()

    With Options
        .AllowBackgroundSave = False
    End With

    ActiveDocument.Save

End Sub
```

OrganizeInFolder Property

Returns or sets a **Boolean** value that specifies whether a Web publication will be saved in a flat structure or hierarchical structure. If **False**, all files in the Web publication will be saved in a flat structure within the root folder. If **True**, the files will be saved in a hierarchical structure within the root folder. The default value is **True**. Read/write.

expression.**OrganizeInFolder**

expression Required. An expression that returns a **WebOptions** object.

Example

The following example specifies that all files in the Web publication should be saved in a flat structure within the root folder.

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    .OrganizeInFolder = False
End With
```


Show All

Orientation Property

 Orientation property as it applies to the **TextFrame** object.

Returns or sets a **PbTextOrientation** constant that represents how text flows in a text box. Read/write.

PbTextOrientation can be one of these PbTextOrientation constants.
pbTextOrientationHorizontal
pbTextOrientationMixed
pbTextOrientationRightToLeft
pbTextOrientationVerticalEastAsia

expression.**Orientation**

expression Required. An expression that returns one of the objects in the Applies To list.

 Orientation property as it applies to the **PageSetup** object.

Returns or sets a **PbOrientationType** constant that specifies whether the page is in landscape or portrait orientation. Read/write.

PbOrientationType can be one of these PbOrientationType constants.
pbOrientationLandscape
pbOrientationPortrait

expression.**Orientation**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

As it applies to the **TextFrame** object.

This example sets the text orientation in the specified text box to vertical so that text flows from top to bottom. This assumes there is at least one shape on page one of the active publication.

```
Sub SetVerticalTextBox()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .Orientation = pbTextOrientationVerticalEastAsia
End Sub
```

As it applies to the **PageSetup** object.

The following example sets the orientation of the pages in the active document to landscape.

```
With ActiveDocument.PageSetup
    .Orientation = pbOrientationLandscape
End With
```

OriginalColorsInPalette Property

Returns a **Long** that represents the number of colors in the specified linked picture's palette. Read-only.

expression.**OriginalColorsInPalette**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures or OLE objects that are not TrueColor (that is, they contain color data of less than 24 bits per channel.) Returns "Permission Denied" for shapes representing embedded or pasted pictures and OLE objects, or linked pictures that are TrueColor.

Use either of the following properties to determine whether a shape represents a linked picture:

- The **Type** property of the **Shape** object
- The **IsLinked** property of the **PictureFormat** object

Use the **OriginalIsTrueColor** property to determine whether a linked picture contains color data of 24 bits per channel or greater.

Example

The following example returns a list of all pictures in the active publication that are not TrueColor. The number of colors in each picture's palette is returned, and if the picture is linked and the linked picture is not TrueColor, the number of colors in its palette is also returned.

```
Sub PictureColorInformation()
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Or shpLoop.Type = pbPictur

            With shpLoop.PictureFormat
                If .IsEmpty = msoFalse Then

                    If .IsTrueColor = msoFalse Then
                        Debug.Print .Filename
                        Debug.Print "This picture has " & .ColorsInP
                        If .IsLinked = msoTrue Then
                            If .OriginalIsTrueColor = msoFalse Then
                                Debug.Print "The linked picture has
                                .OriginalColorsInPalette & " colors.
                            End If
                        End If
                    End If

                End If
            End With

        End If
    Next shpLoop
Next pgLoop

End Sub
```

OriginalFileSize Property

Returns a **Long** representing the size, in bytes, of the linked picture or OLE object. Read-only.

expression.**OriginalFileSize**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

Use either of the following properties to determine whether a shape represents a linked picture:

- The **Type** property of the **Shape** object
- The **IsLinked** property of the **PictureFormat** object

Example

The following example tests each picture in the active publication, and prints selected image properties for pictures that are linked.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat

                        Debug.Print "File Name: " & .Filename
                        Debug.Print "Original File Size: " & .Origin

            End With
        End If
    Next shpLoop
Next pgLoop
```


Show All

OriginalHasAlphaChannel Property

Returns an **MsoTriState** constant depending on whether the original, linked picture contains an alpha channel. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The original, linked picture does not contain an alpha channel.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The original, linked picture contains an alpha channel.

expression.**OriginalHasAlphaChannel**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

Use either of the following properties to determine whether a shape represents a linked picture:

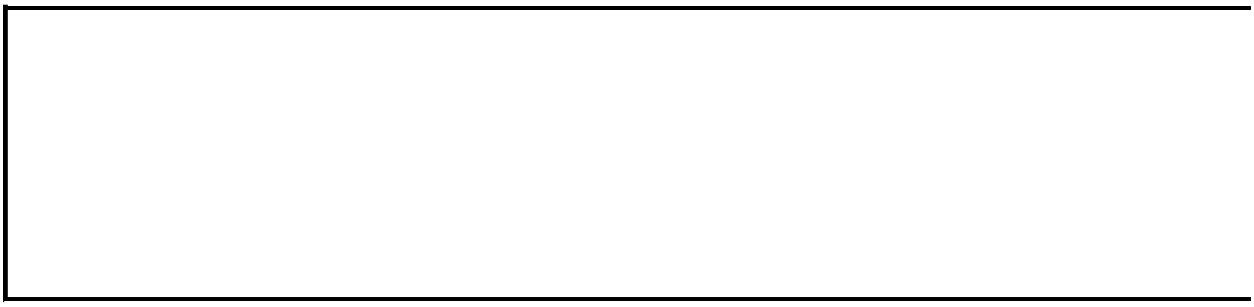
- The **Type** property of the **Shape** object
- The **IsLinked** property of the **PictureFormat** object

An alpha channel is a special 8-bit channel used by some image processing software to contain additional data, such as masking or transparency information.

Example

The following example returns whether the first shape on the first page of the active publication contains an alpha channel. If the picture is linked, and the original picture contains an alpha channel, that is also returned. This example assumes the shape is a picture.

```
With ActiveDocument.Pages(1).Shapes(1).PictureFormat
    If .HasAlphaChannel = msoTrue Then
        Debug.Print .Filename
        Debug.Print "This picture contains an alpha channel."

        If .IsLinked = msoTrue Then
            If .OriginalHasAlphaChannel = msoTrue Then
                Debug.Print "The linked picture " & _
                "also contains an alpha channel."
            End If
        End If
    End If
End With
```



OriginalHeight Property

Returns a **Variant** representing the height, in points, of the specified linked picture or OLE object. Read-only.

expression.**OriginalHeight**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures or OLE objects. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Example

The following example tests each picture in the active publication, and returns selected image properties for pictures that are linked.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat

                        Debug.Print "File Name: " & .Filename
                        Debug.Print "Horizontal Scaling: " & .Horizo
                        Debug.Print "Original Image Height: " & .Ori
                        Debug.Print "Height in publication: " & .Hei

            End With
        End If
    Next shpLoop
Next pgLoop
```


Show All

OriginalIsTrueColor Property

Returns an **MsoTriState** constant indicating whether the specified linked picture or OLE object contains color data of 24 bits per channel or greater. Read-only.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The specified linked picture does not contain color data of 24 bits per channel or greater.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The specified linked picture contains color data of 24 bits per channel or greater.

expression.**OriginalIsTrueColor**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures or OLE objects. Returns "Permission Denied" for shapes representing embedded or pasted pictures and OLE objects.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Example

The following example returns a list of pictures in the active document that are TrueColor. If a picture is linked, and the linked picture is also TrueColor, that information is also returned.

```
Sub PictureColorInformation()
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Or shpLoop.Type = pbPictur

            With shpLoop.PictureFormat
                If .IsEmpty = msoFalse Then

                    If .IsTrueColor = msoTrue Then
                        Debug.Print .Filename
                        Debug.Print "This picture is TrueColor"
                        If .IsLinked = msoTrue Then
                            If .OriginalIsTrueColor = msoTrue Then
                                Debug.Print "The linked picture is a
                            End If
                        End If
                    End If

                End If
            End With

        End If
    Next shpLoop
Next pgLoop

End Sub
```

OriginalResolution Property

Returns a **Long** that represents, in dots per inch (dpi), the resolution at which the linked picture was originally scanned. Read-only.

expression.**OriginalResolution**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Use the **EffectiveResolution** property to determine the resolution at which the picture or OLE object will print in the specified document.

Example

The following example tests each picture in the active publication, and returns selected image properties for pictures that are linked.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat

                        Debug.Print "File Name: " & .Filename
                        Debug.Print "Resolution in Publication: " &
                        Debug.Print "Original Resolution: " & .Origi

            End With
        End If
    Next shpLoop
Next pgLoop
```

OriginalWidth Property

Returns a **Variant** that represents, in points, the width of the specified linked picture or OLE object. Read-only.

expression.**OriginalWidth**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

This property only applies to linked pictures. Returns "Permission Denied" for shapes representing embedded or pasted pictures.

To determine whether a shape represents a linked picture, use either the **Type** property of the **Shape** object, or the **IsLinked** property of the **PictureFormat** object.

Example

The following example tests each picture in the active publication, and returns selected image properties for pictures that are linked.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbLinkedPicture Then

            With shpLoop.PictureFormat

                        Debug.Print "File Name: " & .Filename
                        Debug.Print "Vertical Scaling: " & .Vertical
                        Debug.Print "Original Image Width: " & .Orig
                        Debug.Print "Width in publication: " & .Widt

            End With
        End If
    Next shpLoop
Next pgLoop
```


Show All

Outline Property

Returns or sets an **MsoTriState** constant that represents the state of the outline formatting property on the characters in the specified text range. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters have outline formatting.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse**.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All characters in the range have outline formatting.

expression.**Outline**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests all the text in the second story of the active publication, and if it has mixed outline formatting, it removes all outline formatting. If all or none of the text is formatted as outline, a message box is displayed telling the user that outlining is not mixed.

```
Sub OutlineStory()

    Dim stf As Font

    Set stf = Application.ActiveDocument.Stories(2).TextRange.Font
    With stf
        If .Outline = msoTriStateMixed Then
            .Outline = msoFalse
        Else
            MsgBox "Outlining is not mixed in this story."
        End If
    End With

End Sub
```


Show All

Overflowing Property

MsoTrue if the text frame contains more text than can fit into the text frame. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse

msoTriStateMixed Not used with this property.

msoTriStateToggle Not used with this property.

msoTrue

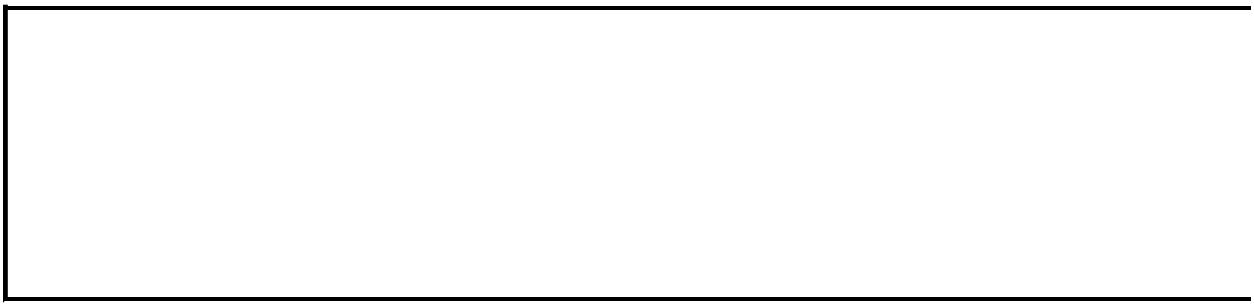
expression.**Overflowing**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example increases the height of the selected text frame if it contains overflowing text.

```
Sub IncreaseTextBoxHeight()
    With Selection.ShapeRange.TextFrame
        If .Overflowing = msoTrue Then
            Do
                .Parent.Height = .Parent.Height + 36
            Loop Until .Overflowing = msoFalse
        End If
    End With
End Sub
```



PageCount Property

Returns a **Long** indicating the number of pages in the specified reader spread. Read-only.

expression.**PageCount**

expression Required. An expression that returns one of the objects in the Applies To list.

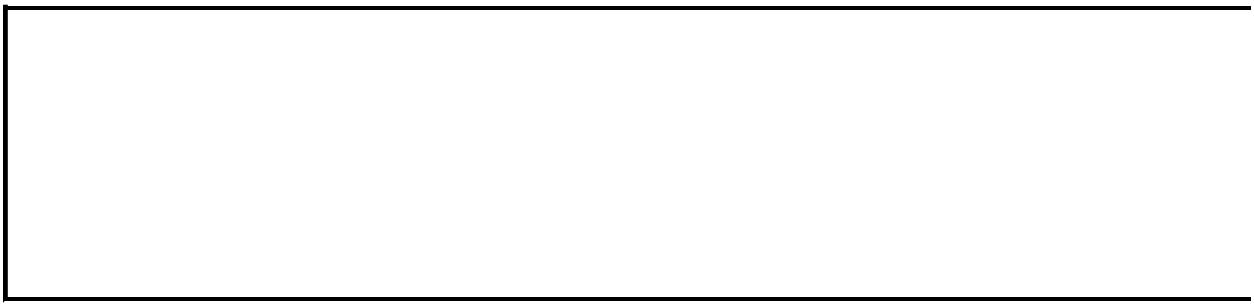
Remarks

A reader spread can contain no more than two pages.

Example

The following example checks the reader spread of the third page in the active publication to see if it contains more than one page, then displays the total number of pages in the spread.

```
Sub NumberOfPagesInSpread()
    If ActiveDocument.Pages(3).ReaderSpread.PageCount > 1 Then
        MsgBox "The spread has two pages."
    Else
        MsgBox "The spread has only one page."
    End If
End Sub
```



PageHeight Property

Returns or sets a **Variant** that represents the height of the pages in a publication. Read/write.

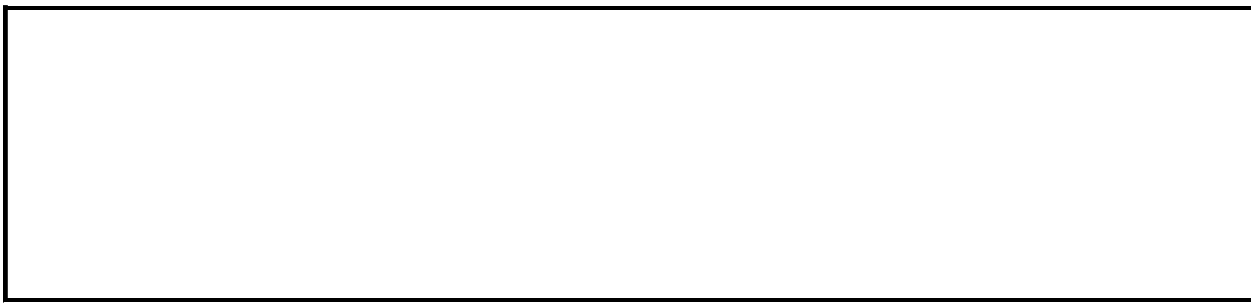
expression.**PageHeight**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example specifies a height of five inches for the pages in the active publication.

```
Sub SetLeftMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```




Show All

PageID Property

 PageID property as it applies to the **Hyperlink** object.

Returns or sets a **Long** indicating the page in the publication that is the destination for the specified hyperlink. Read/write.

expression.**PageID**

expression Required. An expression that returns one of the above objects.

 PageID property as it applies to the **Page** object.

Returns a **Long** indicating the unique identifier for a page in a publication. Read-only.

expression.**PageID**

expression Required. An expression that returns one of the above objects.

Remarks

PageID values are random numbers assigned to pages when they are added. These unique numbers do not change when pages are added or deleted. Also, these numbers do not start with 1, nor are they contiguous.

Example

 As it applies to the **Hyperlink** object.

The following example looks at the first hyperlink in the active publication and reports what page it is linked to.

```
Dim hypTemp As Hyperlink
Dim lngID As Long
Dim strPage As String

Set hypTemp = ActiveDocument.Pages(1).Shapes(1).Hyperlink

lngID = hypTemp.PageID
strPage = ActiveDocument.Pages.FindByPageID(PageID:=lngID).PageNumbe

MsgBox "This hyperlink goes to the page " & strPage & "."
```

 As it applies to the **Page** object.

The following example displays the **PageIndex**, **PageNumber**, and **PageID** properties for all the pages in the active publication.

```
Dim lngLoop As Long

With ActiveDocument.Pages
    For lngLoop = 1 To .Count
        With .Item(lngLoop)
            Debug.Print "PageIndex = " & .PageIndex _
                & " / PageNumber = " & .PageNumber _
                & " / PageID = " & .PageID
        End With
    Next lngLoop
End With
```

PageIndex Property

Returns a **Long** indicating the ordinal number of a page within its publication. Read-only.

expression.**PageIndex**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

A **PageIndex** property value is assigned to each page when it is added, and the value is incremented for each additional page. When pages are added or deleted, page index numbers are reassigned such that the first page is always 1 and the page index numbers always increment by 1. For example, in a publication with five pages, the page index numbers will be 1 through 5, regardless of the page number displayed on the pages themselves.

Example

The following example displays the **PageIndex**, **PageNumber**, and **PageID** properties for all the pages in the active publication.

```
Dim lngLoop As Long

With ActiveDocument.Pages
For lngLoop = 1 To .Count
With .Item(lngLoop)
Debug.Print "PageIndex = " & .PageIndex _
& " / PageNumber = " & .PageNumber _
& " / PageID = " & .PageID
End With
Next lngLoop
End With
```

PageNumber Property

Returns a **String** that represents the current page number. Read-only.

expression.**PageNumber**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a text box, gets the current page number, and inserts it with new text into the new shape.

```
Sub GetPageNumber()
    Dim strPageNumber As String
    With ActiveDocument.Pages(1)
        strPageNumber = .PageNumber
        .Shapes.AddTextbox(Orientation:=pbTextOrientationHorizontal,
            Left:=100, Top:=100, Width:=100, Height:=100) _
            .TextFrame.TextRange.InsertAfter NewText:="This is page
            & strPageNumber & " of " & .Parent.Count & "."
    End With
End Sub
```


Show All

PageNumberFormat Property

Sets or returns a **pbPageNumberFormat** constant that reperesents the formatting of the page numbering. Read/write.

pbPageNumberFormat can be one of these **pbPageNumberFormat** constants.

pbPageNumberFormatAiueo
pbPageNumberFormatArabic
pbPageNumberFormatArabic1
pbPageNumberFormatArabic2
pbPageNumberFormatArabicLZ
pbPageNumberFormatCardtext
pbPageNumberFormatChnDbNum2
pbPageNumberFormatChnDbNum3
pbPageNumberFormatChosung
pbPageNumberFormatCirclenum
pbPageNumberFormatDAiueo
pbPageNumberFormatDbChar
pbPageNumberFormatDbNum1
pbPageNumberFormatDbNum2
pbPageNumberFormatDbNum3
pbPageNumberFormatDIroha
pbPageNumberFormatGanada
pbPageNumberFormatHebrew1
pbPageNumberFormatHebrew2
pbPageNumberFormatHindi1
pbPageNumberFormatHindi2
pbPageNumberFormatHindi3
pbPageNumberFormatHindi4
pbPageNumberFormatIroha
pbPageNumberFormatKorDbNum1

pbPageNumberFormatKorDbNum2
pbPageNumberFormatKorDbNum3
pbPageNumberFormatKorDbNum4
pbPageNumberFormatLCLetter
pbPageNumberFormatLCRoman
pbPageNumberFormatLCRus
pbPageNumberFormatOrdinal
pbPageNumberFormatOrdtext
pbPageNumberFormatThai1
pbPageNumberFormatThai2
pbPageNumberFormatThai3
pbPageNumberFormatTpeDbNum2
pbPageNumberFormatTpeDbNum3
pbPageNumberFormatTpeDbNum3
pbPageNumberFormatUCLetter
pbPageNumberFormatUCRoman
pbPageNumberFormatUCRus
pbPageNumberFormatViet1
pbPageNumberFormatZodiac1
pbPageNumberFormatZodiac2

expression.**PageNumberFormat**

expression Required. An expression that returns a **Section** object.

Remarks

Not all of the **pbPageNumberFormat** constants will be available depending on the languages that are enabled or installed.

Example

This example adds a new section to the active document, sets the page number format to lower case roman, and then sets the starting page number to 1.

```
Dim objSection As Section
Set objSection = ActiveDocument.Sections.Add(2)
With objSection
    .PageNumberFormat = pbPageNumberFormatLCRoman
    .PageNumberStart = 1
End With
```

PageNumberStart Property

Sets or returns the page number that the specified section starts with. Read/write **Long**.

expression.**PageNumberStart**

expression Required. An expression that returns a **Section** object.

Example

The following example sets the starting page number for the first section of the active document to 45.

```
ActiveDocument.Sections(1).PageNumberStart = 45
```


Show All

Pages Property

 Pages property as it applies to the **Document** object.

Returns a **Pages** collection representing all the pages in the specified publication.

expression.**Pages**

expression Required. An expression that returns one of the above objects.

 Pages property as it applies to the **ReaderSpread** object.

Returns a **Page** object representing one of the pages that comprise the specified reader spread.

expression.**Pages**(***Index***)

expression Required. An expression that returns one of the above objects.

Index Required **Long**. The page from the reader spread to return. Can be either 1 or 2.

Remarks

A reader spread will consist of only one or two pages, hence the valid values for the ***Index*** argument.

Example

 As it applies to the **Document** object.

The following example returns the **Pages** collection of the active publication and reports how many pages there are.

```
Dim pgsTemp As Pages

Set pgsTemp = ActiveDocument.Pages

With pgsTemp
    MsgBox "There are " & .Count _
        & " page(s) in the active publication."
End With
```

 As it applies to the **ReaderSpread** object.

The following example checks the reader spread of the fifth page in the active publication to see if it contains more than one page. If it does, the example reports the page number of the second page in the spread.

```
Dim pageTemp As Page

With ActiveDocument.Pages(5).ReaderSpread
    If .PageCount > 1 Then
        Set pageTemp = .Pages(Index:=2)
        MsgBox "The page number of the second page " _
            & "in the spread is " & pageTemp.PageNumber
    Else
        MsgBox "The spread has only one page."
    End If
End With
```

PageSetup Property

Returns a **PageSetup** object representing a publication’s page size, page layout and paper settings. Read-only.

expression.**PageSetup**

expression Required. An expression that returns one of the objects in the Applies To list.

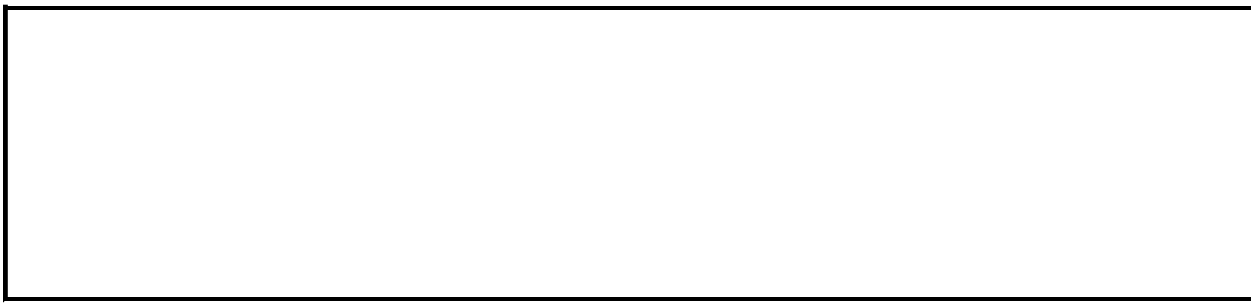
Remarks

You can only use the **PageSetup** property when printing multiple pages on a single sheet of printer paper. If the page size is greater than half the paper size, Publisher will display an error.

Example

This example specifies page setup options for a publication with multiple publication pages printed on each sheet of printer paper.

```
Sub SetTopMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .TopMargin = InchesToPoints(0.25)
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```




Show All

PageType Property

Returns a **PbPageType** constant that represents the page type. Read-only.

PbPageType can be one of these PbPageType constants.
pbPageLeftPage
pbPageMasterPage
pbPageRightPage
pbPageScratchPage

expression.**PageType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a shape on alternating corners of each page in the active publication.

```
Sub GetPageType()
    Dim pgCount As Page
    For Each pgCount In ActiveDocument.Pages
        If pgCount.PageType = pbPageLeftPage Then
            pgCount.Shapes.AddShape Type:=msoShapeOval, _
                Left:=50, Top:=50, Width:=50, Height:=50
        ElseIf pgCount.PageType = pbPageRightPage Then
            pgCount.Shapes.AddShape Type:=msoShapeOval, _
                Left:=512, Top:=50, Width:=50, Height:=50
        End If
    Next
End Sub
```

PageWidth Property

Returns or sets a **Variant** that represents the width of the pages in a publication. Read/write.

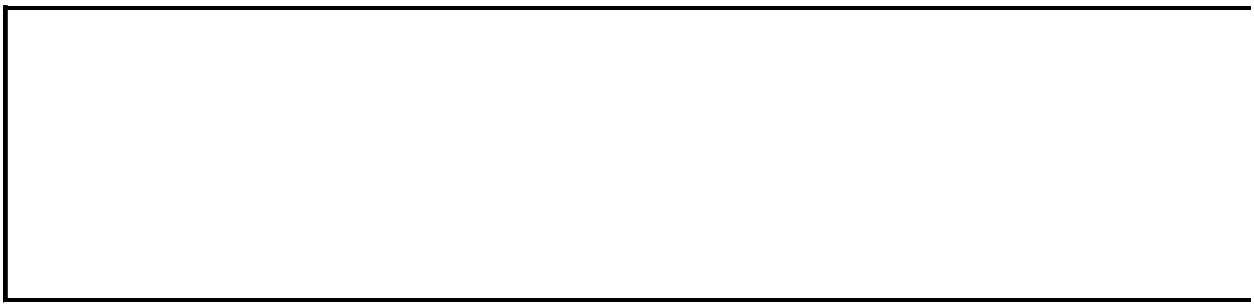
expression.**PageWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example specifies a width of eight inches for the pages in the active publication.

```
Sub SetLeftMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```



ParagraphFormat Property

Returns a **ParagraphFormat** object representing the paragraph formatting for the specified text range or text style.

expression.**ParagraphFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example removes all the tab stops from the text in the first shape on page one of the active publication.

```
Dim pfTemp As ParagraphFormat

Set pfTemp = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.ParagraphFormat

pfTemp.Tabs.ClearAll
```

Parent Property

Returns an object that represents the parent object of the specified object. For example, for a **TextFrame** object, returns a **Shape** object representing the parent shape of the text frame. Read-only.

expression.**Parent**

expression Required. An expression that returns one of the objects in the Applies To list.

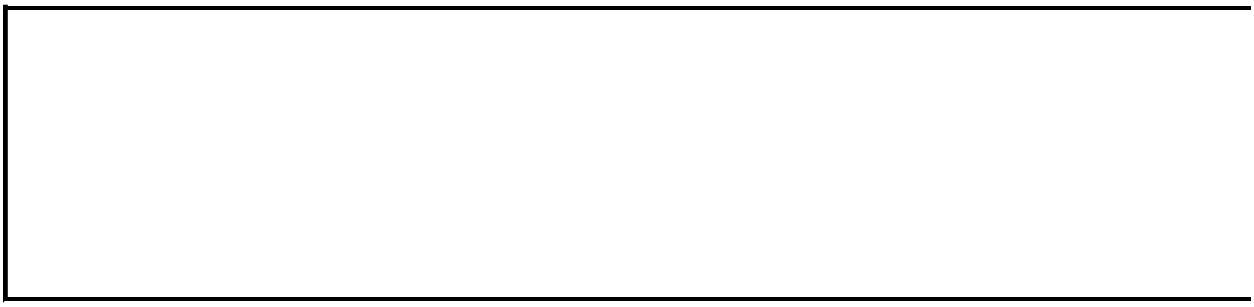
Example

This example accesses the parent object of the selected shape, and then adds a new shape to it and sets the fill for the new shape.

```
Sub ParentObject()
    Dim shp As Shape
    Dim pg As Page

    Set pg = Selection.ShapeRange(1).Parent
    Set shp = pg.Shapes.AddShape(Type:=msoShape5pointStar, _
        Left:=72, Top:=72, Width:=72, Height:=72)

    shp.Fill.ForeColor.RGB = RGB(Red:=180, Green:=180, Blue:=180)
End Sub
```

This example returns the parent object of a text frame, which is the first shape in the active publication, and then fills the shape with a pattern.

```
Sub ParentShape()
    Dim shpParent As Shape
    Set shpParent = ActiveDocument.Pages(1).Shapes(1).TextFrame.Pare
    shpParent.Fill.Patterned Pattern:=msoPatternSphere
End Sub
```



ParentGroupShape Property

Returns a **Shape** object that represents the common parent shape of a child shape or a range of child shapes.

expression.**ParentGroupShape**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two shapes in the active document and groups those shapes. Then using one shape in the group, it accesses the parent group and fills all shapes in the parent group with the same fill pattern. This example assumes that the active document does not currently contain any shapes. If it does, an error may occur.

```
Sub ParentGroupShape()
    Dim shpGroup As Shape

    With ActiveDocument.Pages(1).Shapes
        .AddShape Type:=msoShapeOval, Left:=72, _
            Top:=72, Width:=100, Height:=100
        .AddShape Type:=msoShapeHeart, Left:=110, _
            Top:=120, Width:=100, Height:=100
        .Range(Array(1, 2)).Group
    End With

    Set shpGroup = ActiveDocument.Pages(1).Shapes(1) _
        .GroupItems(1).ParentGroupShape
    shpGroup.Fill.Patterned Pattern:=msoPattern25Percent

End Sub
```

Path Property

Returns a **String** indicating the full path to the file of the saved active publication, not including the last separator or file name. Read-only.

expression.**Path**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **FullName** property can be used to return both the path and file name.

Example

The following example demonstrates the differences between the **Path**, **Name**, and **FullName** properties. This example is best illustrated if the publication is saved in a folder other than the default.

```
Sub PathNames()

    Dim strPath As String
    Dim strName As String
    Dim strFullName As String

    strPath = Application.ActiveDocument.Path
    strName = Application.ActiveDocument.Name
    strFullName = Application.ActiveDocument.FullName

    ' Note the file name & path differences
    ' while executing.
    MsgBox "The path is: " & strPath
    MsgBox "The file name is: " & strName
    MsgBox "The path & file name are: " & strFullName

End Sub
```

PathForPictures Property

Returns or sets a **String** that represents the default path for picture files. Read/write.

expression.**PathForPictures**

expression Required. An expression that returns one of the objects in the Applies To list.

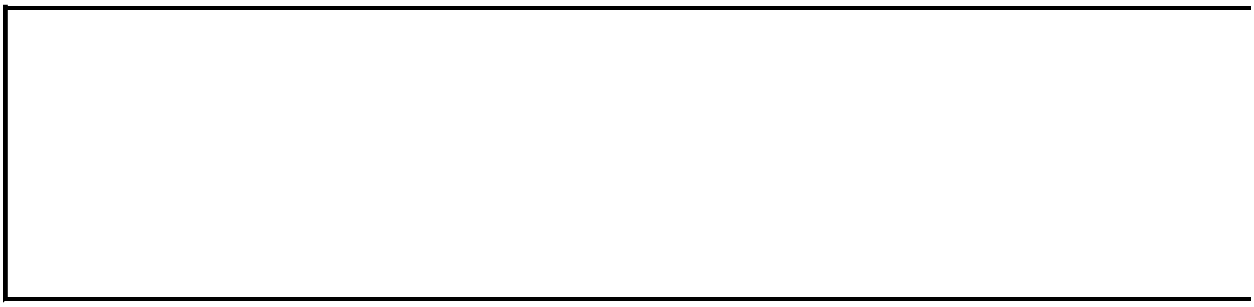
Example

This example sets the path for picture files. (Note that *PathToFolder* must be replaced with a valid folder path for this example to work.)

```
Sub SetPicturePath()
    Options.PathForPictures = "PathToFolder"
End Sub
```

This example places the default path for picture files in a string and then uses the path string to add the specified file to the active publication. (Note that *Filename* must be replaced with a valid file name for this example to work.)

```
Sub InsertNewPicture()
    Dim strPicPath As String

    strPicPath = Options.PathForPictures

    ActiveDocument.Pages(1).Shapes.AddPicture FileName:=strPicPath _
        & "Filename", LinktoFile:=msoFalse, _
        SaveWithDocument:=msoTrue, Left:=50, Top:=50, Height:=200

End Sub
```



PathForPublications Property

Returns or sets a **String** that represents the default folder for publications. Read/write.

expression.**PathForPublications**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The new setting takes effect immediately.

Example

This example sets the default folder for Publisher documents. (Note that *PathToFolder* must be replaced with a valid folder path for this example to work.)

```
Sub ChangeDefaultPath()
    Options.PathForPublications = "PathToFolder"
End Sub
```

This example returns the current default path for publications (corresponds to the default path setting on the **General** tab in the **Options** dialog box, **Tools** menu).

```
Sub PubPath()
    Dim strPubPath
    strPubPath = Options.PathForPublications
    MsgBox strPubPath
End Sub
```

PathSeparator Property

Returns a **String** that represents the character used to separate folder names. Read-only.

expression.**PathSeparator**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

You can use **PathSeparator** to build Web addresses even though they contain forward slashes (/).

The **FullName** property returns the path and file name as a single string.

For worldwide compatibility, it is recommended to use this property when building paths rather than referring explicitly to path separator characters in code (for example, "/").

Example

This example displays the path and file name of the active document.

```
Sub PathFileName()

    With Application
        MsgBox "The name of the active document: " & vbLf & _
            .Path & .PathSeparator & ActiveDocument.Name
    End With

End Sub
```


Show All

Pattern Property

Returns or sets an **MsoPatternType** constant that represents the pattern applied to the specified fill or line. Read-only for the **FillFormat** object; read/write for the **LineFormat** object.

MsoPatternType can be one of these MsoPatternType constants.

msoPattern10Percent
msoPattern20Percent
msoPattern25Percent
msoPattern30Percent
msoPattern40Percent
msoPattern50Percent
msoPattern5Percent
msoPattern60Percent
msoPattern70Percent
msoPattern75Percent
msoPattern80Percent
msoPattern90Percent
msoPatternDarkDownwardDiagonal
msoPatternDarkHorizontal
msoPatternDarkUpwardDiagonal
msoPatternDarkVertical
msoPatternDashedDownwardDiagonal
msoPatternDashedHorizontal
msoPatternDashedUpwardDiagonal
msoPatternDashedVertical
msoPatternDiagonalBrick
msoPatternDivot
msoPatternDottedDiamond
msoPatternDottedGrid
msoPatternHorizontalBrick

msoPatternLargeCheckerBoard
msoPatternLargeConfetti
msoPatternLargeGrid
msoPatternLightDownwardDiagonal
msoPatternLightHorizontal
msoPatternLightUpwardDiagonal
msoPatternLightVertical
msoPatternMixed
msoPatternNarrowHorizontal
msoPatternNarrowVertical
msoPatternOutlinedDiamond
msoPatternPlaid
msoPatternShingle
msoPatternSmallCheckerBoard
msoPatternSmallConfetti
msoPatternSmallGrid
msoPatternSolidDiamond
msoPatternSphere
msoPatternTrellis
msoPatternWave
msoPatternWeave
msoPatternWideDownwardDiagonal
msoPatternWideUpwardDiagonal
msoPatternZigZag

expression.**Pattern**

expression Required. An expression that returns one of the above objects.

Example

This example sets the pattern for the specified shape if the shape currently doesn't have a fill pattern. This example assumes that at least one shape exists on the first page of the active publication.

```
Sub ChangeFillPattern()
    With ActiveDocument.Pages(1).Shapes(1).Fill
        If .Pattern < msoPattern10Percent Then
            .Patterned Pattern:=msoPattern25Percent
        End If
    End With
End Sub
```


Show All

PersonalInformationSet Property

Returns or sets a **PbPersonalInfoSet** constant indicating the current identity set. Read/write.

PbPersonalInfoSet can be one of these PbPersonalInfoSet constants.

pbPersonalInfoHome
pbPersonalInfoOtherOrganization
pbPersonalInfoPrimaryBusiness
pbPersonalInfoSecondaryBusiness

expression.**PersonalInformationSet**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

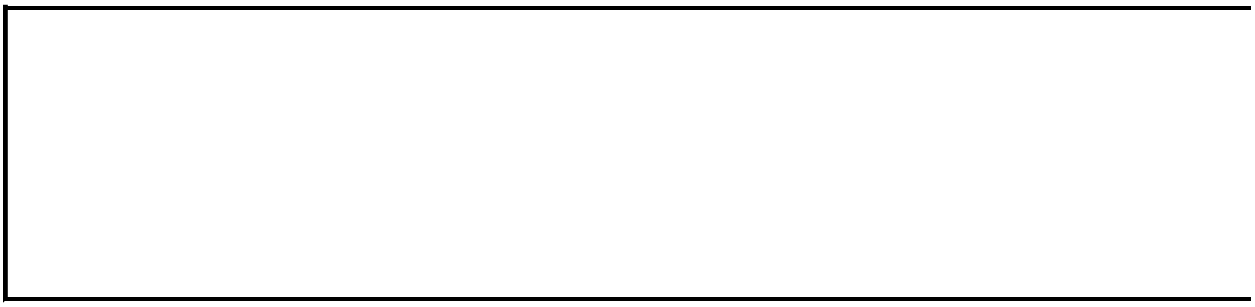
Setting this property will change all the identity information in the publication.

Caution Use this property with caution. Sensitive or confidental information could be revealed to other users.

Example

The following statement sets the current publication's identity information to home information.

```
Application.ActiveDocument.PersonalInformationSet = pbPersonalInfoHo
```




Show All

Perspective Property

MsoTrue if the extrusion appears in perspective— that is, if the walls of the extrusion narrow toward a vanishing point. **MsoFalse** if the extrusion is a parallel, or orthographic, projection— that is, if the walls don't narrow toward a vanishing point. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**Perspective**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the extrusion depth for shape one on the first page to 100 points and specifies that the extrusion be parallel, or orthographic. For this example to work, the specified shape must be a 3-D shape.

```
Sub ChangePerspective()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        .Visible = True
        .Depth = 100
        .Perspective = msoFalse
    End With
End Sub
```

PhoneticGuide Property

Returns a **PhoneticGuide** object that represents the properties of phonetic text applied to a text range.

expression.**PhoneticGuide**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds phonetic text to the selection and displays the text to which the phonetic text applies, which is the originally selected text. This example assumes text is selected. If no text is selected, the message box will be blank.

```
Sub AddPhoneticText()
    With Selection.TextRange.Fields.AddPhoneticGuide _
            (Range:=Selection.TextRange, Text:="ver-E nIs")
        MsgBox "The base text is " & .PhoneticGuide.BaseText
    End With
End Sub
```

PictureFormat Property

Returns a **PictureFormat** object that contains picture formatting properties for the specified object. Applies to **Shape** or **ShapeRange** objects that represent pictures or OLE objects.

expression.**PictureFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the brightness and contrast for all pictures on the first page of the active publication.

```
Sub FixPictureContrastBrightness()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        If shp.Type = pbPicture Then
            With shp.PictureFormat
                .Brightness = 0.6
                .Contrast = 0.6
            End With
        End If
    Next shp
End Sub
```

Plates Property

Returns a **Plates** collection representing the color plates for the specified publication.

expression.**Plates**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example returns the plates collection for the active publication and lists the names of all the color plates.

```
Dim plaTemp As Plates
Dim plaLoop As Plate

Set plaTemp = ActiveDocument.Plates

If ActiveDocument.ColorMode = pbColorModeDesktop Then
    Debug.Print "Desktop color mode: No color plates!"
Else
    For Each plaLoop In plaTemp
        Debug.Print "The name of this plate is " & plaLoop.Name
    Next plaLoop
End If
```


Show All

Points Property

Returns a **Variant** that represents the position of the specified node as a coordinate pair. Each coordinate is expressed in points. Read-only.

expression.**Points**

expression Required. An expression that returns one of the objects in the Applies To list.

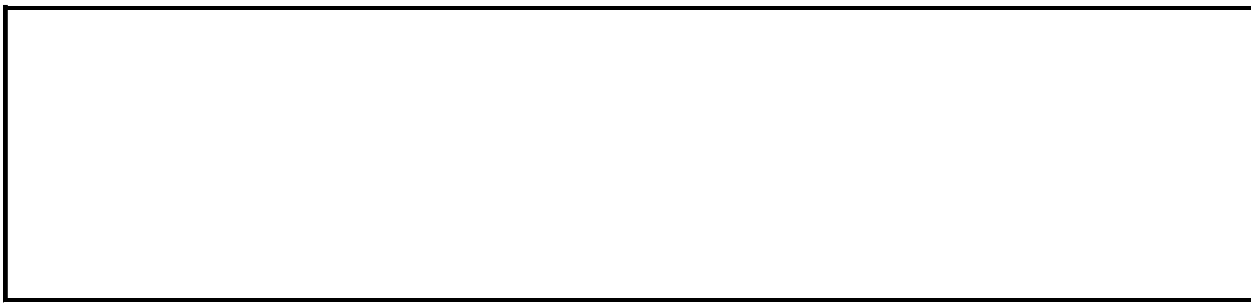
Remarks

This property is read-only. Use the **SetPosition** method to set the location of the node.

Example

This example moves node two in shape one on the first page of the active publication to the right 200 points and down 300 points. For this example to work, shape one must be a freeform drawing.

```
Sub SetPointsPosition()
    Dim varArray As Variant
    Dim intX As Integer
    Dim intY As Integer
    With ActiveDocument.Pages(1).Shapes(1).Nodes
        varArray = .Item(2).Points
        intX = varArray(1, 1)
        intY = varArray(1, 2)
        .SetPosition Index:=2, X1:=intX + 200, Y1:=intY + 300
    End With
End Sub
```



Position Property

Returns or sets a **Variant** representing the font position relative to the baseline of the text in the specified range. Positive values move the text above the normal baseline, negative values move the text below the baseline. Indeterminate values are returned as -9999.0. Read/write.

expression.**Position**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Numeric values are evaluated in points; string values can be in any measurement units supported by Microsoft Publisher (for example, "0.25 in").

Example

This example adjusts the text in the second story to 5 points below the baseline.

```
Sub Position()

    Application.ActiveDocument.Stories(2).TextRange.Font.Position =

End Sub
```


Show All

PostFormData Property

Returns or sets an **MsoTriState** constant indicating whether the specified Web command button control uses the **Get** or **Post** method when submitting form data to a Web server. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The control uses the Visual Basic **Get** method to submit form data.

msoTriStateMixed Not used with this property.

msoTriStateToggle Not used with this property.

msoTrue *default* The control uses the Visual Basic **Post** method to submit form data.

expression.**PostFormData**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is ignored for Reset command buttons.

Example

This example creates a Web form Submit command button and sets the script path and file name to run when a user clicks the button. The example also specifies that the Web form should use the Visual Basic **Get** method to submit form data.

```
Dim shpNew As Shape

Set shpNew = ActiveDocument.Pages(1).Shapes.AddWebControl _
    (Type:=pbWebControlCommandButton, Left:=150, _
    Top:=150, Width:=75, Height:=36)

With shpNew.WebCommandButton
    .ButtonText = "Submit"
    .ButtonType = pbCommandButtonSubmit
    .ActionURL = "http://www.tailspintoys.com/" _
        & "scripts/ispscript.cgi"
    .PostFormData = msoFalse
End With
```


Show All

PresetExtrusionDirection Property

Returns an **MsoPresetExtrusionDirection** constant that represents the direction taken by the extrusion's sweep path leading away from the extruded shape (the front face of the extrusion). Read-only.

MsoPresetExtrusionDirection can be one of these MsoPresetExtrusionDirection constants.

msoExtrusionBottom
msoExtrusionBottomLeft
msoExtrusionBottomRight
msoExtrusionLeft
msoExtrusionNone
msoExtrusionRight
msoExtrusionTop
msoExtrusionTopLeft
msoExtrusionTopRight
msoPresetExtrusionDirectionMixed

expression.**PresetExtrusionDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is read-only. To set the value of this property, use the **SetExtrusionDirection** method.

Example

This example changes the extrusion for the first shape on the first page of the active publication if the extrusion extends toward the upper-left corner of the extrusion's front face. For this example to work, the specified shape must be a 3-D shape.

```
Sub SetExtrusion()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        If .PresetExtrusionDirection = msoExtrusionTopLeft Then
           .SetExtrusionDirection msoExtrusionBottomRight
        End If
    End With
End Sub
```


Show All

PresetGradientType Property

Returns an **MsoPresetGradientType** that represents the preset gradient type for the specified fill. Read-only.

MsoPresetGradientType can be one of these MsoPresetGradientType constants.

msoGradientBrass
msoGradientCalmWater
msoGradientChrome
msoGradientChromeII
msoGradientDaybreak
msoGradientDesert
msoGradientEarlySunset
msoGradientFire
msoGradientFog
msoGradientGold
msoGradientGoldII
msoGradientHorizon
msoGradientLateSunset
msoGradientMahogany
msoGradientMoss
msoGradientNightfall
msoGradientOcean
msoGradientParchment
msoGradientPeacock
msoGradientRainbow
msoGradientRainbowII
msoGradientSapphire
msoGradientSilver
msoGradientWheat
msoPresetGradientMixed

expression.**PresetGradientType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **PresetGradient** method to set the preset gradient type for the fill.

Example

This example changes the fill for the first shape on the first page of the active publication to the Fog preset gradient fill if it is not already set to the Fog preset gradient. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub SetGradient()
    With ActiveDocument.Pages(1).Shapes(1).Fill
        If .PresetGradientType <> msoGradientFog Then
            .PresetGradient Style:=msoGradientHorizontal, _
                Variant:=1, PresetGradientType:=msoGradientFog
        End If
    End With
End Sub
```


Show All

PresetLightingDirection Property

Returns or sets an **MsoPresetLightingDirection** constant that represents the position of the light source relative to the extrusion. Read/write.

MsoPresetLightingDirection can be one of these MsoPresetLightingDirection constants.

msoLightingBottom
msoLightingBottomLeft
msoLightingBottomRight
msoLightingLeft
msoLightingNone
msoLightingRight
msoLightingTop
msoLightingTopLeft
msoLightingTopRight
msoPresetLightingDirectionMixed

expression.**PresetLightingDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The lighting effects you set won't be apparent if the extrusion has a wire frame surface.

Example

This example sets the extrusion for the first shape on the first page of the active publication to extend toward the top of the shape and that the lighting for the extrusion come from the left. For this example to work, the specified shape must be a 3-D shape.

```
Sub ExtrusionLighting()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        .Visible = True
        .SetExtrusionDirection msoExtrusionTop
        .PresetLightingDirection = msoLightingLeft
    End With
End Sub
```


Show All

PresetLightingSoftness Property

Returns or sets a **MsoPresetLightingSoftness** constant that represents the intensity of the extrusion lighting. Read/write.

MsoPresetLightingSoftness can be one of these MsoPresetLightingSoftness constants.

msoLightingBright

msoLightingDim

msoLightingNormal

msoPresetLightingSoftnessMixed

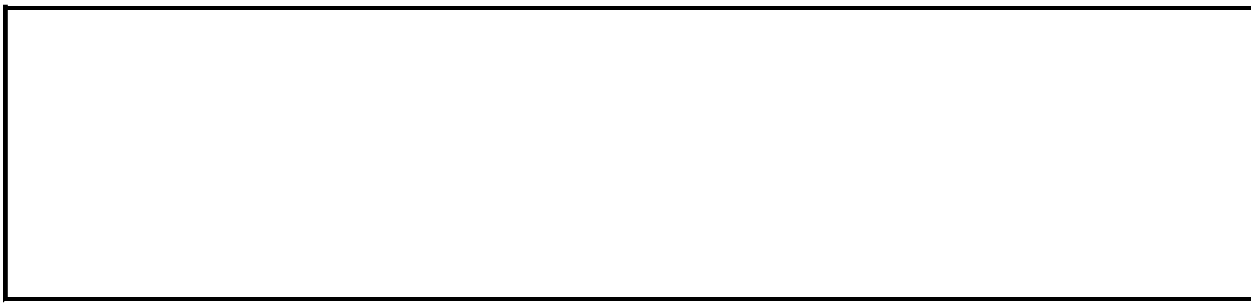
expression.**PresetLightingSoftness**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the extrusion for the first shape on the first page of the active publication to be lit brightly from the left. For this example to work, the specified shape must be a 3-D shape.

```
Sub SetExtrusionLightingBrighness()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        .Visible = True
        .PresetLightingSoftness = msoLightingBright
        .PresetLightingDirection = msoLightingLeft
    End With
End Sub
```




Show All

PresetMaterial Property

Returns or sets an **MsoPresetMaterial** constant that represents the extrusion surface material. Read/write.

MsoPresetMaterial can be one of these MsoPresetMaterial constants.
msoMaterialMatte
msoMaterialMetal
msoMaterialPlastic
msoMaterialWireFrame
msoPresetMaterialMixed

expression.**PresetMaterial**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example specifies that the extrusion surface for shape one in the active publication be a wire frame. For this example to work, the specified shape must be a 3-D shape.

```
Sub SetExtrusionMaterial()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        .Visible = True
        .PresetMaterial = msoMaterialWireFrame
    End With
End Sub
```


Show All

PresetShape Property

Returns or sets an **MsoPresetTextEffectShape** constant that represents the shape of the specified WordArt. Read/write.

MsoPresetTextEffectShape can be one of these MsoPresetTextEffectShape constants.

msoTextEffectShapeArchDownCurve
msoTextEffectShapeArchDownPour
msoTextEffectShapeArchUpCurve
msoTextEffectShapeArchUpPour
msoTextEffectShapeButtonCurve
msoTextEffectShapeButtonPour
msoTextEffectShapeCanDown
msoTextEffectShapeCanUp
msoTextEffectShapeCascadeDown
msoTextEffectShapeCascadeUp
msoTextEffectShapeChevronDown
msoTextEffectShapeChevronUp
msoTextEffectShapeCircleCurve
msoTextEffectShapeCirclePour
msoTextEffectShapeCurveDown
msoTextEffectShapeCurveUp
msoTextEffectShapeDeflate
msoTextEffectShapeDeflateBottom
msoTextEffectShapeDeflateInflate
msoTextEffectShapeDeflateInflateDeflate
msoTextEffectShapeDeflateTop
msoTextEffectShapeDoubleWave1
msoTextEffectShapeDoubleWave2
msoTextEffectShapeFadeDown
msoTextEffectShapeFadeLeft

msoTextEffectShapeFadeRight
msoTextEffectShapeFadeUp
msoTextEffectShapeInflate
msoTextEffectShapeInflateBottom
msoTextEffectShapeInflateTop
msoTextEffectShapeMixed
msoTextEffectShapePlainText
msoTextEffectShapeRingInside
msoTextEffectShapeRingOutside
msoTextEffectShapeSlantDown
msoTextEffectShapeSlantUp
msoTextEffectShapeStop
msoTextEffectShapeTriangleDown
msoTextEffectShapeTriangleUp
msoTextEffectShapeWave1
msoTextEffectShapeWave2

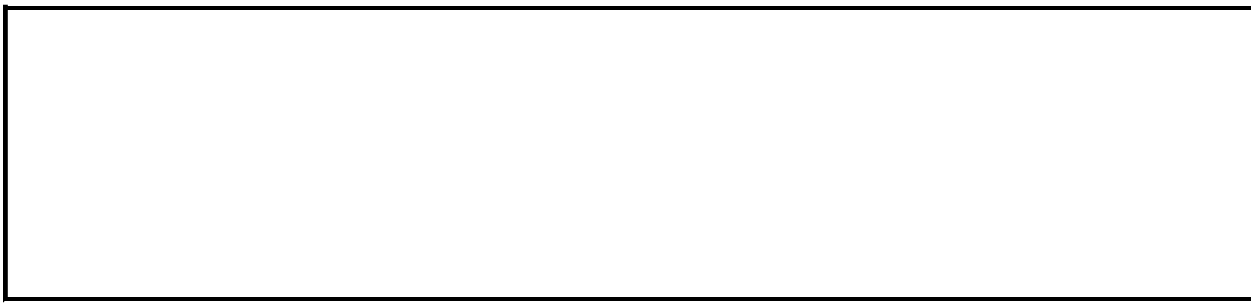
expression.**PresetShape**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the shape of the first shape on the first page of the active publication to a chevron whose center points down. For this example to work the first shape must be a WordArt shape.

```
Sub ChangeTextEffect()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Type = msoTextEffect Then
            .TextEffect.PresetShape = msoTextEffectShapeChevronDown
        End If
    End With
End Sub
```




Show All

PresetTextEffect Property

Returns or sets an **MsoPresetTextEffect** constant that represents the style of the specified WordArt. The values for this property correspond to the formats in the **WordArt Gallery** dialog box, numbered from left to right, top to bottom. Read/write.

MsoPresetTextEffect can be one of these MsoPresetTextEffect constants.

msoTextEffect1
msoTextEffect10
msoTextEffect11
msoTextEffect12
msoTextEffect13
msoTextEffect14
msoTextEffect15
msoTextEffect16
msoTextEffect17
msoTextEffect18
msoTextEffect19
msoTextEffect2
msoTextEffect20
msoTextEffect21
msoTextEffect22
msoTextEffect23
msoTextEffect24
msoTextEffect25
msoTextEffect26
msoTextEffect27
msoTextEffect28
msoTextEffect29
msoTextEffect3
msoTextEffect30

msoTextEffect4
msoTextEffect5
msoTextEffect6
msoTextEffect7
msoTextEffect8
msoTextEffect9
msoTextEffectMixed

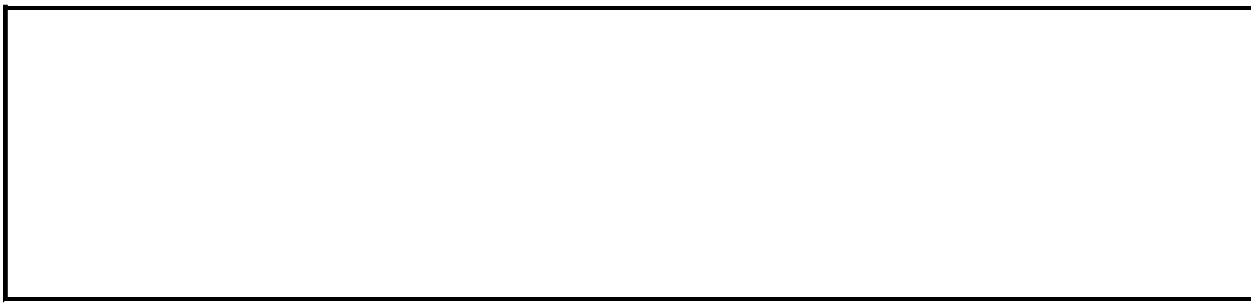
expression.**PresetTextEffect**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the text effect style for the first shape on the first page of the active publication. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub ChangeTextEffect()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Type = msoTextEffect Then
            .TextEffect.PresetTextEffect = msoTextEffect1
        End If
    End With
End Sub
```




Show All

PresetTexture Property

Returns an **MsoPresetTexture** constant that represents the preset texture for the specified fill. Read-only.

MsoPresetTexture can be one of these MsoPresetTexture constants.

msoPresetTextureMixed
msoTextureBlueTissuePaper
msoTextureBouquet
msoTextureBrownMarble
msoTextureCanvas
msoTextureCork
msoTextureDenim
msoTextureFishFossil
msoTextureGranite
msoTextureGreenMarble
msoTextureMediumWood
msoTextureNewsprint
msoTextureOak
msoTexturePaperBag
msoTexturePapyrus
msoTextureParchment
msoTexturePinkTissuePaper
msoTexturePurpleMesh
msoTextureRecycledPaper
msoTextureSand
msoTextureStationery
msoTextureWalnut
msoTextureWaterDroplets
msoTextureWhiteMarble
msoTextureWovenMat

expression.**PresetTexture**

expression Required. An expression that returns one of the objects in the Applies To list.

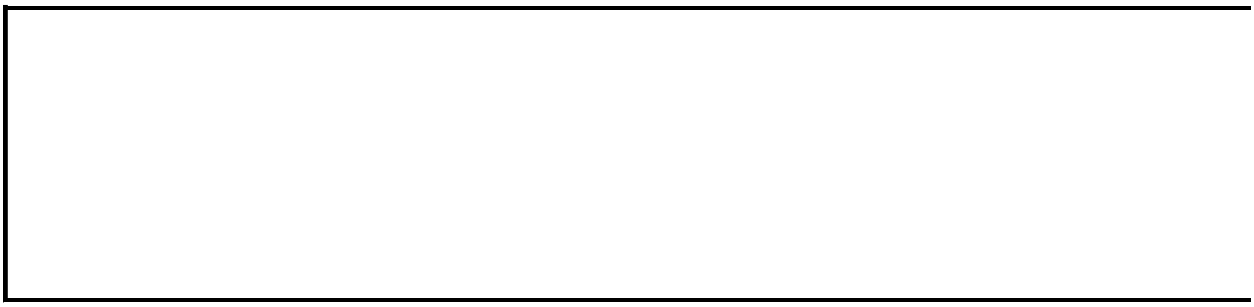
Remarks

Use the **PresetTextured** method to specify the preset texture for the fill.

Example

This example adds a rectangle to the first page in the active publication and sets its preset texture to match that of the first shape on the page. For the example to work, the first shape must have a preset textured fill.

```
Sub SetTexture()
    Dim texture As MsoPresetTexture
    With ActiveDocument.Pages(1).Shapes
            texture = .Item(1).Fill.PresetTexture
        With .AddShape(Type:=msoShapeRectangle, Left:=250, Top:=72,
               Width:=40, Height:=80)
            .Fill.PresetTextured PresetTexture:=texture
        End With
    End With
End Sub
```




Show All

PresetThreeDFormat Property

Returns an **MsoPresetThreeDFormat** constant that represents the preset extrusion format. Each preset extrusion format contains a set of preset values for the various properties of the extrusion. If the extrusion has a custom format rather than a preset format, this property returns **msoPresetThreeDFormatMixed**. Read-only.

MsoPresetThreeDFormat can be one of these MsoPresetThreeDFormat constants.

msoPresetThreeDFormatMixed
msoThreeD1
msoThreeD10
msoThreeD11
msoThreeD12
msoThreeD13
msoThreeD14
msoThreeD15
msoThreeD16
msoThreeD17
msoThreeD18
msoThreeD19
msoThreeD2
msoThreeD20
msoThreeD3
msoThreeD4
msoThreeD5
msoThreeD6
msoThreeD7
msoThreeD8
msoThreeD9

expression.**PresetThreeDFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

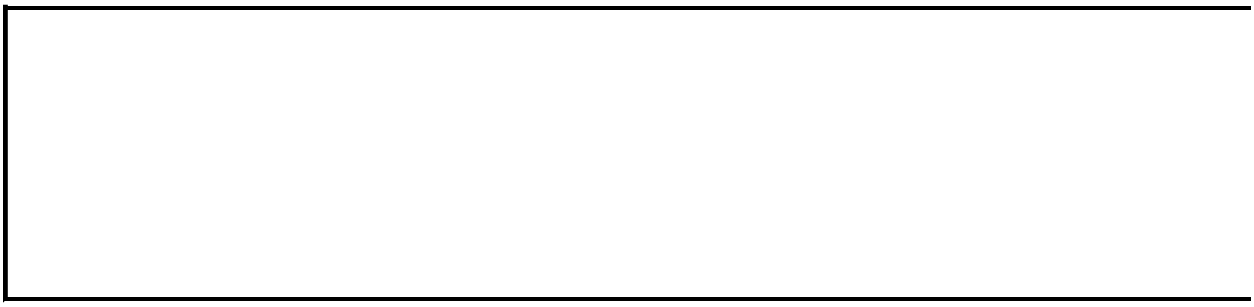
The values for this property correspond to the options (numbered from left to right, top to bottom) displayed when you click the **3-D Style** button on the **Formatting** toolbar.

Use the **SetThreeDFormat** method to set the preset extrusion format.

Example

This example sets the extrusion format for the first shape on the first page of the active publication to 3-D Style 12 if the shape initially has a custom extrusion format. For this example to work, the specified shape must be a 3-D shape.

```
Sub SetPreset3D()
    With ActiveDocument.Pages(1).Shapes(1).ThreeD
        If .PresetThreeDFormat = msoPresetThreeDFormatMixed Then
            .SetThreeDFormat msoThreeD12
        End If
    End With
End Sub
```



PreviousLinkedTextFrame Property

Returns a **TextFrame** object representing the text frame from which text flows to the specified text frame.

expression.**PreviousLinkedTextFrame**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the specified text frame is not part of a chain of linked frames or is the first in a chain of linked frames, this property returns nothing.

Example

The following example returns the previously linked text frame of shape three on page one of the active publication and sets its font to Times New Roman.

```
Dim txtFrame As TextFrame

Set txtFrame = ActiveDocument.Pages(1) _
    .Shapes(3).TextFrame.PreviousLinkedTextFrame

txtFrame.TextRange.Font = "Times New Roman"
```

PrintablePlates Property

Returns a **PrintablePlates** collection. Read-only.

expression.**PrintablePlates**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

The **PrintablePlates** property is only accessible if the publication is set to print as separations. Returns "Permission Denied" if any other print mode is specified.

The **PrintablePlates** collection is generated when a publication's print mode is set to separations. The **PrintablePlates** collection represents the plates that will actually be printed for the publication, based on:

- The plates (if any) you have defined for the publication
- The advanced print options specified

Example

The following example returns all the printable plates currently defined for the active publication, and lists selected properties of each. This example assumes that the print mode of the active publication is set to print separations.

```
Sub ListPrintablePlates()
    Dim pplTemp As PrintablePlates
    Dim pplLoop As PrintablePlate


    Set pplTemp = ActiveDocument.AdvancedPrintOptions.PrintablePlate
    Debug.Print "There are " & pplTemp.Count & " printable plates in

    For Each pplLoop In pplTemp
        With pplLoop
            Debug.Print "Printable Plate Name: " & .Name
            Debug.Print "Index: " & .Index
            Debug.Print "Ink Name: " & .InkName
            Debug.Print "Plate Angle: " & .Angle
            Debug.Print "Plate Frequency: " & .Frequency
            Debug.Print "Print Plate?: " & .PrintPlate
        End With
    Next pplLoop
End Sub
```

PrintableRect Property

Returns a **PrintableRect** object that represents the printer sheet area within which the specified printer will print. Read-only.

expression.**PrintableRect**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

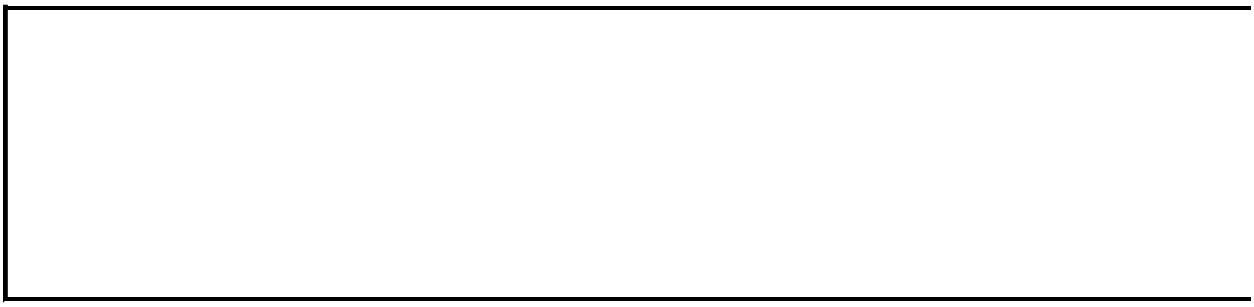
The printable rectangle is determined by the printer based on the sheet size specified. The printable rectangle of the printer sheet should not be confused with the area within the margins of the publication page; it may be larger or smaller than the publication page.

Note In cases in which the printer sheet and the publication page size are identical, the publication page is centered on the printer sheet and none of the printer's marks print, even if they are selected.

Example

The following example returns printable rectangle boundaries for the printer sheet of the active publication.

```
Sub ListPrintableRectBoundaries()

With ActiveDocument.AdvancedPrintOptions.PrintableRect

    Debug.Print "Printable area is " & _
            PointsToInches(.Width) & _
            " by " & PointsToInches(.Height) & " inches."
    Debug.Print "Left Boundary: " & PointsToInches(.Left) & _
                    " inches (from left)."
    Debug.Print "Right Boundary: " & PointsToInches(.Left + .Width)
                    " inches (from left)."
    Debug.Print "Top Boundary: " & PointsToInches(.Top) & _
                    " inches(from top)."
    Debug.Print "Bottom Boundary: " & PointsToInches(.Top + .Height)
                    " inches(from top)."

End With

End Sub
```



PrintBlankPlates Property

False to prevent printing plates when an ink is used within a document, but not on a specific page. For example, suppose a document contains red and black spot colors on first page, but the rest of the pages contain black only. If **PrintBlankPlates** is set to **False**, a red plate will be printed for the first page, but not for any of the following pages because they do not contain red. The default is **True**. Read/write **Boolean**.

expression.**PrintBlankPlates**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property is only accessible if separations are being printed. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specifiy that separations are to be printed. Returns "Permission Denied" if any other print mode is specified.

This property corresponds to the **Don't print blank plates** control on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

The following example tests to determine if the active publication has been set to print as separations. If it has, it is set to print only plates for the inks actually used in the publication, and to not print plates for any pages where a color is not used.

```
Sub PrintOnlyInksUsed
        With ActiveDocument.AdvancedPrintOptions
                If .PrintMode = pbPrintModeSeparations Then
                        .InksToPrint = pbInksToPrintUsed
                        .PrintBlankPlates = False
                End If
        End With
End Sub
```

PrintBleedMarks Property

True to print bleed marks in the specified publication. The default is **False**. Read/write **Boolean**.

expression.**PrintBleedMarks**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Bleed marks show the extent of a bleed, and print an eighth inch outside the crop marks.

This property is only accessible if bleeds are allowed in the specified publication. Use the **AllowBleeds** property of the **AdvancedPrintOptions** object to specify bleeds are allowed. Returns "Permission Denied" if bleeds are not allowed in the publication.

This property corresponds to the **Bleed marks** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

Example

The following example sets the publication to allow bleeds, and to print bleed marks.

```
Sub AllowBleedsAndPrintMarks()
        With ActiveDocument.AdvancedPrintOptions
                .AllowBleeds = True
                .PrintBleedMarks = True
        End With
End Sub
```

PrintCMYKByDefault Property

True to use composite CMYK as the default print mode for future Publisher instances, rather than composite RGB. Read/write **boolean**.

expression.**PrintCMYKByDefault**

expression Required. An expression that returns a **AdvancedPrintOptions** object.

Remarks

Setting this property to **True** sets the default value of the **PrintMode** property for future instances of Publisher to **pbPrintModeCompositCMYK**. Setting this property does not apply to the current application of Publisher, or any open instances.

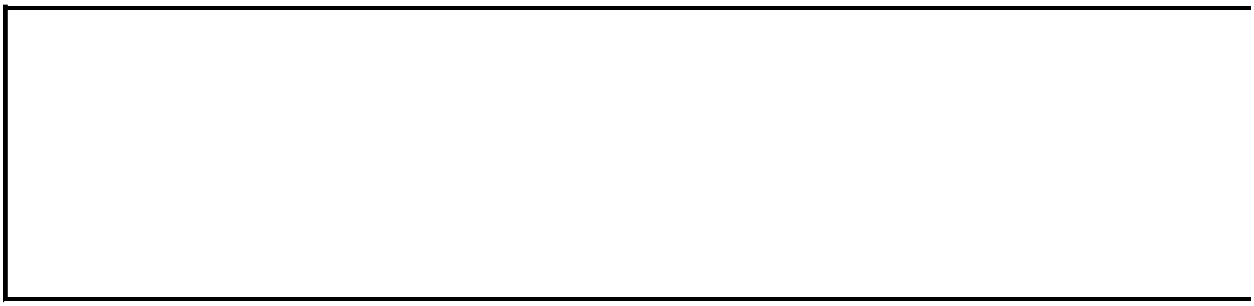
Use the **PrintMode** property of the **AdvancedPrintOptions** object to specify a publication's print mode. The default print mode value is **pbPrintModeCompositeRGB**.

This property corresponds to the **Print Composite CMYK by default** check box on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

The following example sets the default print mode of future instances of Publisher to composite CMYK.

```
ActiveDocument.AdvancedPrintOptions.PrintCMYKByDefault = True
```



PrintColorBars Property

True to print a color bar for the specified publication. The default is **True**. Read/write **Boolean**.

expression.**PrintColorBars**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Returns "Permission Denied" if any print mode other than separations is selected for the specified publication. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specifiy the print mode for a publication.

Color bars are used by commercial printers to determine how a solid patch of ink prints on the page.

This property corresponds to the **Color bars** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

These printer's marks print outside of the publication and can only be printed if the size of the paper being printed to is larger than the publication page size.

Example

The following example sets crop marks and job information to print with the publication. If the publication is printed as separations, the additional types of printer's marks are also set to print. This example assumes that the size of the paper being printed to is larger than the publication page size.

```
Sub SetPrintersMarksToPrint()
        With ActiveDocument.AdvancedPrintOptions
                .PrintCropMarks = True
                .PrintJobInformation = True
                        If PrintMode = pbPrintModeSeparations Then
                                .PrintRegistrationMarks = True
                                .PrintDensityBars = True
                                .PrintColorBars = True
                        End If
                End With
End Sub
```

PrintCropMarks Property

True to print crop marks for the specified publication. The default is **True**. Read/write **Boolean**.

expression.**PrintCropMarks**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property corresponds to the **Crop marks** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

Crop marks are used as guides when a printed publication is trimmed to its intended size.

These printer's marks print outside of the publication and can only be printed if the size of the sheet being printed to is larger than the publication page size.

Example

The following example sets crop marks and job information to print with the publication. If the publication is printed as separations, the additional types of printer's marks are also set to print. This example assumes that the size of the paper being printed to is larger than the publication page size.

```
Sub SetPrintersMarksToPrint()
        With ActiveDocument.AdvancedPrintOptions
                .PrintCropMarks = True
                .PrintJobInformation = True
                        If PrintMode = pbPrintModeSeparations Then
                                .PrintRegistrationMarks = True
                                .PrintDensityBars = True
                                .PrintColorBars = True
                        End If
                End With
End Sub
```

PrintDensityBars Property

True to print a density bar for the specified publication. The default is **True**. Read/write **Boolean**.

expression.**PrintDensityBars**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Returns "Permission Denied" if any print mode other than separations is selected for the specified publication. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specifiy the print mode for a publication.

The density bar printed is graded from a 10 percent screen to a 100 percent fill. A commercial printer can use this bar to determine proper exposure time for plate burning, and to test dot gain in the printed pages.

This property corresponds to the **Density bars** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

These printer's marks print outside of the publication and can only be printed if the size of the paper being printed on is larger than the publication page size.

Example

The following example sets crop marks and job information to print with the publication. If the publication is printed as separations, the additional types of printer's marks are also set to print. This example assumes that the size of the paper being printed to is larger than the publication page size.

```
Sub SetPrintersMarksToPrint()
        With ActiveDocument.AdvancedPrintOptions
                .PrintCropMarks = True
                .PrintJobInformation = True
                        If PrintMode = pbPrintModeSeparations Then
                                .PrintRegistrationMarks = True
                                .PrintDensityBars = True
                                .PrintColorBars = True
                        End If
                End With
End Sub
```

PrintJobInformation Property

True to print information about the print job on each plate. The default is **True**. Read/write **Boolean**.

expression.**PrintJobInformation**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

The **PrintJobInformation** property can be set regardless of the print mode selected for the publication. However, it is ignored (and no job information is printed) when the print mode is set as composite RGB. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specifiy the print mode for a publication.

Job information comprises the file name of the printed publication, the date it was printed, the page number, and which color ink the plate is for (cyan, magenta, yellow, black, or a spot color).

This property corresponds to the **Job information** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

These printer's marks print outside of the publication and can only be printed if the size of the paper being printed to is larger than the publication page size.

Example

The following example sets crop marks and job information to print with the publication. If the publication is printed as separations, the additional types of printer's marks are also set to print. This example assumes that the size of the paper being printed to is larger than the publication page size.

```
Sub SetPrintersMarksToPrint()
        With ActiveDocument.AdvancedPrintOptions
                .PrintCropMarks = True
                .PrintJobInformation = True
                        If PrintMode = pbPrintModeSeparations Then
                                .PrintRegistrationMarks = True
                                .PrintDensityBars = True
                                .PrintColorBars = True
                        End If
                End With
End Sub
```

PrintLineByLine Property

Returns or sets a **Boolean** indicating whether to print documents line by line (applies to inkjet printers only); **True** to print line by line. Read/write.

expression.**PrintLineByLine**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets Publisher to print line by line on an inkjet printer.

```
Options.PrintLineByLine = True
```


Show All

PrintMode Property

Returns or sets a **PbPrintMode** constant that represents whether the specified publication is printed as a composite or separations. Read/write.

PbPrintMode can be one of these PbPrintMode constants.

pbPrintModeCompositeCMYK Print a composite whose colors are defined by the CMYK color model.

pbPrintModeGrayscale Print a composite whose colors are defined as shades of gray.

pbPrintModeCompositeRGB Print a composite whose colors are defined by the RGB color model.

pbPrintModeSeparations Print a separate plate for each ink used in the publication.

expression.**PrintMode**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

The **PrintMode** property applies to the publication only as it is currently being printed. This property is not saved with the publication, nor as an application setting.

The default value for the **PrintMode** property is **pbPrintModeCompositeRGB**. To specify **pbPrintModeCompositeCMYK** as the default value for future instances of Publisher, use the **PrintCMYKByDefault** property.

This property corresponds to the **Output** control on the **Separations** tab of the **Advanced Print Settings** dialog box.

The **PrintablePlates** collection is generated when a publication's print mode is set to separations. The PrintablePlates collection represents the plates that will actually be printed for the publication, based on:

- The plates (if any) you have defined for the publication
- The advanced print options specified

If you specify separations as the print mode, you can specify which plates to print by using the **InksToPrint** property of the **AdvancedPrintOptions** object. You can also prevent printing plates for any pages where a color is not used by setting the **PrintBlankPlates** property.

Example

The following example tests to determine if the active publication has been set to print as separations. If it has, it is set to print only plates for the inks actually used in the publication, and to not print plates for any pages where a color is not used.

```
Sub PrintOnlyInksUsed
        With ActiveDocument.AdvancedPrintOptions
                If .PrintMode = pbPrintModeSeparations Then
                        .InksToPrint = pbInksToPrintUsed
                        .PrintBlankPlates = False
                End If
        End With
End Sub
```

PrintPageBackgrounds Property

Returns or sets **True** to include page backgrounds when printing pages from the specified publication. Default is **True**. Read/write **Boolean**.

expression.**PrintPageBackgrounds**()

expression Required. An expression that returns a **Document** object.

Remarks

Use the **PageBackground** object to create, alter, or delete the background of a specified page.

Example

The following example sets page backgrounds to print for the active publication.

```
ActiveDocument.PrintPageBackgrounds = True
```

PrintPreview Property

True to display in Print Preview the publication in the current view. Read/write **Boolean**.

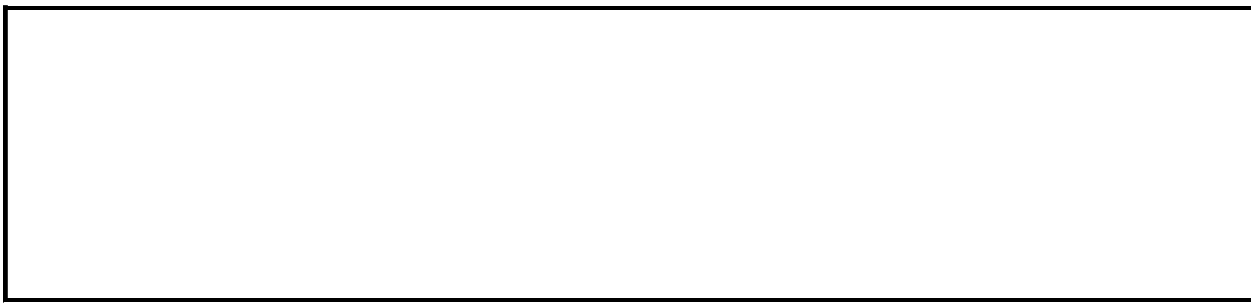
expression.**PrintPreview**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example switches the view to Print Preview.

```
Sub GoToPrintPreview()
    PrintPreview = True
End Sub
```



PrintRegistrationMarks Property

True to print registration marks for the specified publication. The default is **True**. Read/write **Boolean**.

expression.**PrintRegistrationMarks**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Returns "Permission Denied" if any print mode other than separations is selected for the specified publication. Use the **PrintMode** property of the **AdvancedPrintOptions** object to specify the print mode for a publication.

This property corresponds to the **Registration marks** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

Registration marks are used to align (register) the printing of two or more press plates on a single page.

These printer's marks print outside of the publication and can only be printed if the size of the paper being printed to is larger than the publication page size.

Example

The following example sets crop marks and job information to print with the publication. If the publication is printed as separations, the additional types of printer's marks are also set to print. This example assumes that the size of the paper being printed to is larger than the publication page size.

```
Sub SetPrintersMarksToPrint()
        With ActiveDocument.AdvancedPrintOptions
                .PrintCropMarks = True
                .PrintJobInformation = True
                        If PrintMode = pbPrintModeSeparations Then
                                .PrintRegistrationMarks = True
                                .PrintDensityBars = True
                                .PrintColorBars = True
                        End If
                End With
End Sub
```

ProductCode Property

Returns a **String** indicating the Microsoft Publisher globally unique identifier (GUID). Read-only.

expression.**ProductCode**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the product code for Microsoft Publisher.

```
MsgBox "The product code for Microsoft Publisher is " _
    & ProductCode
```


Show All

ProgId Property

Returns a **String** that represents the programmatic identifier (ProgID) for the specified OLE object. Read-only.

expression.**ProgId**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example loops through all the linked OLE object shapes on the first page of the active document and updates all linked Excel worksheets. This example assumes there is at least one shape on the first page of the active publication.

```
Sub UpdateLinkedOLEObject()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        If shp.Type = msoLinkedOLEObject Then
            If shp.OLEFormat.ProgId = "Excel.Sheet" Then
                shp.LinkFormat.Update
            End If
        End If
    Next
End Sub
```

Properties Property

Returns a **WizardProperties** collection representing all the settings that are part of the specified publication design or Design Gallery object's wizard.

expression.**Properties**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example reports on the publication design associated with the active publication, displaying its name and current settings.

```
Dim wizTemp As Wizard
Dim wizproTemp As WizardProperty
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp
    Set wizproAll = .Properties
    Debug.Print "Publication Design associated with " _
        & "current publication: " _
        & .Name
    For Each wizproTemp In wizproAll
        With wizproTemp
            Debug.Print "   Wizard property: " _
                & .Name & " = " & .CurrentValueId
        End With
    Next wizproTemp
End With
```

Note Depending on the language version of Publisher that you are using, you may receive an error when using the above code. If this occurs, you will need to build in error handlers to circumvent the errors. For more information, see **Wizard Object**.


Show All

PublicationLayout Property

Returns or sets a **PbPublicationLayout** constant that indicates the layout of a publication. Read/write.

PbPublicationLayout can be one of these PbPublicationLayout constants.

pbLayout4x6BaePan
pbLayout4x6BanPan
pbLayout4x6Pan
pbLayoutBannerCustom
pbLayoutBannerLarge
pbLayoutBannerMedium
pbLayoutBannerSmall
pbLayoutBook
pbLayoutBusinessCardEurope
pbLayoutBusinessCardFE
pbLayoutBusinessCardLocal
pbLayoutBusinessCardUS
pbLayoutCrownPan
pbLayoutCustom
pbLayoutEnvelope
pbLayoutFoldCard
pbLayoutFullPage
pbLayoutGreetingCardL
pbLayoutGreetingCardT
pbLayoutIndexCard
pbLayoutJang4x6Pan
pbLayoutKookBaePan
pbLayoutKookBanPan
pbLayoutKookPan
pbLayoutLabel
pbLayoutPostcardA4

pbLayoutPostcardHalfLetter
pbLayoutPostcardJapan
pbLayoutPostcardUS
pbLayoutPosterLarge
pbLayoutPosterSmall
pbLayoutShinKookPan
pbLayoutShinSeoPan
pbLayoutWebPageLarge
pbLayoutWebPageSmall

expression.**PublicationLayout**

expression Required. An expression that returns a **PageSetup** object.

Remarks

Using the **PublicationLayout** property to set the layout of a publication is identical to setting the layout from the listbox in the Page Setup dialog.

Example

The following example sets the layout of the active publication to **pbLayoutBusinessCardUS**, which by default specifies a page width of 3.5 inches and a page height of 2 inches.

```
With ActiveDocument.PageSetup
    .PublicationLayout = pbLayoutBusinessCardUS
End With
```


Show All

PublicationType Property

Returns a **PbPublicationType** constant that represents the type of the specified publication. Read-only.

PbPublicationType can be one of these pbPublicationType constants.

pbTypePrint

pbTypeWeb

expression.**PublicationType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example determines if the active publication is a print publication. If it is, the publication is converted to a Web publication.

```
Sub ChangePublicationType()
    With ActiveDocument
        If .PublicationType = pbTypePrint Then
        .ConvertPublicationType (pbTypeWeb)
        End If
    End With
End Sub
```

PublishFileName Property

Returns or sets a **String** that represents the file name of a Web page (within a Web publication) that is being saved as filtered HTML.

expression.**PublishFileName**

expression Required. An expression that returns a **WebPageOptions** object.

Remarks

Specifying a file name for a Web page is optional. When a publication is saved as filtered html, Publisher automatically generates file names for any Web page that does not have a file name specified. Use the **SaveAs** method of the **Document** object to save a publication as filtered html.

User defined file names are only used when a publication is saved as filtered html.

File names must be specified without a file extension.

Including invalid characters (such as characters that are not universally allowed in file names that are part of URLs) in the file name will generate a run-time error. Invalid characters include:

- characters with a code point value of below (decimal) 48
- any double-byte characters
- the following symbols: \, ?, >, <, |, : , and .

This property corresponds to the **File name** text box in the **Publish to the Web** section of the **Web Page Options** dialog box.

Example

The following example sets the file name and description of the second page in the active publication. The example assumes the active publication is a web publication with at least two pages.

```
With ActiveDocument.Pages(2).WebPageOptions
    .PublishFileName = "CompanyProfile"
    .Description = "Tailspin Toys Company Profile"
End With
```

Raise Property

Returns a **Variant** indicating the distance between the top of the base text and the bottom of the guide text. Read-only.

expression.**Raise**

expression Required. An expression that returns one of the objects in the Applies To list.

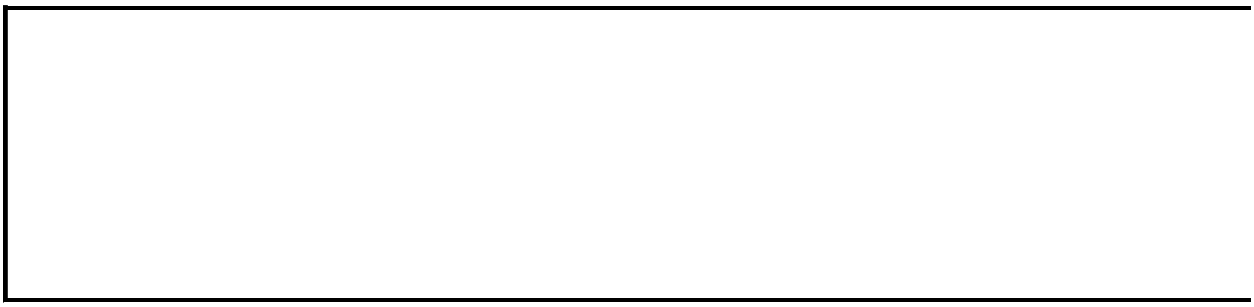
Remarks

Numeric set values are in points; strings can be any measurement unit supported by Microsoft Publisher. Return values are always in points.

Example

The following example places the phonetic guide for shape one in the active publication five points above the base text.

```
Dim phoGuide As PhoneticGuide

Set phoGuide = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Fields(1).PhoneticGuide

With phoGuide
    .Raise = 5
End With
```




Show All

Range Property

 **Range** property as it applies to the **InlineShapes** collection.

Returns a **ShapeRange** collection representing the same set of inline shapes as the **InlineShapes** collection whose method was called. This allows for miscellaneous formatting of the contained shapes. Read-only.

expression.**Range**(***Index***)

expression Required. An expression that returns one of the objects in the Applies To list.

Index Optional **Long**. The index position of the inline shape within the **ShapeRange**.

 **Range** property as it applies to the **Hyperlink** object.

Returns a **TextRange** object representing the base text to which the specified hyperlink has been applied.

expression.**Range**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the **Type** property of the specified **Hyperlink** object is a value other than **msoHyperlinkRange**, the **Range** property returns nothing.

Example

As it applies to the **InlineShapes** collection.

The following example searches through each shape on the first page of the publication, and for all inline shapes within each shape, finds the first inline shape within the range of inline shapes and flips it vertically.

```
Dim theShape As Shape
Dim theShapes As Shapes

Set theShapes = ActiveDocument.Pages(1).Shapes

For Each theShape In theShapes
    With theShape.TextFrame.TextRange
        .InlineShapes.Range(1).Flip (msoFlipVertical)
    End With
Next
```

As it applies to the **Hyperlink** object.

The following example returns the text range associated with the first hyperlink on page one of the active publication and changes the base text to "Go here."

```
Dim txtHyperlink As TextRange

txtHyperlink = ActiveDocument.Pages(1) _
    .Shapes(1).Hyperlink.Range

txtHyperlink.Text = "Go here"
```

ReaderSpread Property

Returns a **ReaderSpread** object that represents the reader spread of the specified page.

expression.**ReaderSpread**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example checks to see if the reader spread for the specified page includes less than two pages. If it does, it changes the reader spread to include two pages.

```
Sub SetFacingPages()
    With ActiveDocument.Pages(2).ReaderSpread
        If .PageCount < 2 Then _
            ActiveDocument.ViewTwoPageSpread = True
    End With
End Sub
```

ReadOnly Property

Returns **True** if the publication is read-only; **False** if it is read-write. Read-only **Boolean**.

expression.**ReadOnly**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example saves the active publication and notifies the user that the file is saved and if it is read-only or not.

```
Sub SaveAndStatus()

    Dim bStatus As Boolean

    Application.ActiveDocument.SaveAs "c:\testfile.pub"
    bStatus = Application.ActiveDocument.ReadOnly
    MsgBox "File Saved and Read-only Status = " & bStatus

End Sub
```

RecordCount Property

Returns a **Long** that represents the number of records in the data source. Read-only.

expression.**RecordCount**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example validates ZIP codes in the attached data source for five digits. If the length of the ZIP code is less than five, the record is excluded from the mail merge process. This example assumes the postal codes are U.S. ZIP codes. You could modify this example to search for ZIP codes that have a 4-digit locator code appended to the ZIP code, and then exclude all records that don't contain the locator code.

```
Sub Validate

    Dim intCount As Integer
    With ActiveDocument.MailMerge.DataSource

        'Set the active record equal to the first included record in
        'data source
        .ActiveRecord = 1
        Do
            intCount = intCount + 1

            'Set the condition that field six must be greater than o
            'equal to five digits in length
            If Len(.DataFields.Item(6).Value) < 5 Then

                'Exclude the record if field six is less than five d
                .Included = False

                'Mark the record as containing an invalid address fi
                .InvalidAddress = True

                'Specify the comment attached to the record explaini
                'why the record was excluded from the mail merge
                .InvalidComments = "The ZIP code for this record is
                    & "less than five digits. It will be removed " _
                    & "from the mail merge process."

            End If

            'Move the record to the next record in the data source
            .ActiveRecord = .ActiveRecord + 1

        'End the loop when the counter variable
        'equals the number of records in the data source
        Loop Until intCount = .RecordCount
    End With
```

RedoActionsAvailable Property

Returns the number of actions available on the redo stack. Read-only **Long**.

expression.**RedoActionsAvailable**

expression Required. An expression that returns a **Document** object.

Example

The following example adds a rectangle that contains a text frame to the fourth page of the active publication. Some font properties and the text of the text frame are set. A test is then run to determine whether the font in the text frame is Courier. If so, the **Undo** method is used with the value of the **UndoActionsAvailable** property passed as a parameter to specify that all previous actions be undone.

The **Redo** method is then used with the value of the **RedoActionsAvailable** property minus 2 passed as a parameter to redo all actions except for the last two. A new font is specified for the text in the text frame, in addition to new text.

This example assumes the active document contains at least four pages.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(4)

With theDoc
    With thePage
        Set theShape = .Shapes.AddShape(msoShapeRectangle, _
            75, 75, 190, 30)
        With theShape.TextFrame.TextRange
             .Font.Size = 12
             .Font.Name = "Courier"
             .Text = "This font is Courier."
        End With
     End With

    If thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Courier" T
        ' The Undo method specifies that all undoable actions be und
        .Undo (.UndoActionsAvailable)
        ' The Redo method uses RedoActionsAvailable - 2 to specify t
        ' all redoable actions be redone except for the last two act
        ' The last two actions that are not redone are setting
        ' .Font.Name and .Text.
        .Redo (.RedoActionsAvailable - 2)
        With theShape.TextFrame.TextRange
            .Font.Name = "Verdana"
            .Text = "This font is Verdana."
```

```
        End With
    End If
End With
```

RelyOnVML Property

Returns or sets a **Boolean** value that specifies whether or not image files are generated from drawing objects when a Web publication is saved. If **True**, image files are not generated. If **False**, images are generated. The default value is **False**. Read/write.

expression.**RelyOnVML**

expression Required. An expression that returns a **WebOptions** object.

Remarks

File sizes can be reduced by not generating images for drawing objects. Note that a Web browser must support Vector Markup Language (VML) in order to render drawing objects. Microsoft Internet Explorer versions 5.0 and later support VML, so the **RelyOnVML** property could be set to **True** if targeting those browsers. For browsers that do not support VML, a drawing object will not appear when a Web publication is saved with this property enabled.

If unsure about which browsers will be used to view the Web site, this property should be set to **False**.

Example

The following example assumes that end users have Microsoft Internet Explorer version 5.0, and therefore specifies that images should not be generated from drawing objects when the Web publication is saved.

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    .RelyOnVML = True
End With
```

RemovePersonalInformation Property

Returns or sets a **Boolean** that represents whether to save personal information when the file is saved. Read/write.

expression.**RemovePersonalInformation**

expression Required. An expression that returns a **Document** object.

Remarks

The information removed from the document is the Author, Manager, Company, and the GUID of the computer on which the document was created.

The default setting for this property is **False**.

Example

This example removes the personal information from the active document.

```
ActiveDocument.RemovePersonalInformation = True
```


Show All

ReplaceScope Property

Sets or returns a **PbReplaceScope** constant that specifies how many replacements are to be made: one, all, or none. Read/write **PbReplaceScope**.

PbReplaceScope can be one of these **PbReplaceScope** constants.
pbReplaceScopeAll
pbReplaceScopeNone
pbReplaceScopeOne

expression.**ReplaceScope**

expression Required. An expression that returns a **FindReplace** object.

Remarks

The default setting of the **ReplaceScope** property is **pbReplaceScopeNone**.

Example

The following example replaces all occurrences of the word "hi" with "hello" in the active document.

```
With ActiveDocument.Find
    .Clear
    .FindText = "hi"
    .ReplaceWithText = "hello"
    .MatchWholeWord = True
    .ReplaceScope = pbReplaceScopeAll
    .Execute
End With
```

ReplaceWithText Property

Sets or retrieves a **String** representing the replacement text in the specified range or selection. Read/write.

expression.**ReplaceWithText**

expression Required. An expression that returns a **FindReplace** object.

Remarks

The default setting of the **ReplaceWithText** property is an empty **String**.

If the **ReplaceScope** property is set to either **pbReplaceScopeOne** or **pbReplaceScopeAll** and the **ReplaceWithText** property is not set, the text found will be replaced with the default empty string, thus removing the text.

Example

The following example replaces all occurrences of the word "hello" with "goodbye" in the active document.

```
With ActiveDocument.Find
    .Clear
    .FindText = "hello"
    .ReplaceWithText = "goodbye"
    .MatchWholeWord = True
    .ReplaceScope = pbReplaceScopeAll
    .Execute
End With
```


Show All

RequiredControl Property

MsoTrue if an entry into a Web text box control is required. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse Indicates entry into the specified Web text box control is not required.

msoTriStateMixed

msoTriStateToggle

msoTrue Indicates entry into the specified Web text box control is required.

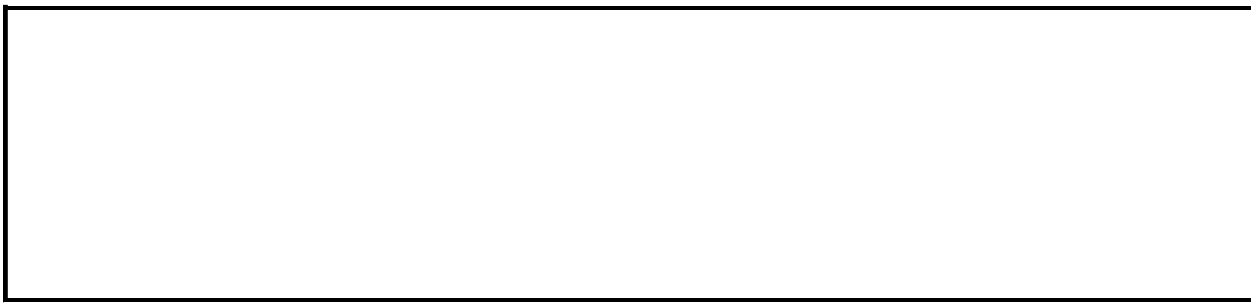
expression.**RequiredControl**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web text box control in the active publication, sets the default text and the character limit for the text box, and specifies that an entry is required.

```
Sub AddWebTextBoxControl()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlMultiLineTextBox, Left:=72, _
            Top:=72, Width:=300, Height:=100).WebTextBox
        .DefaultText = "Please enter text here."
        .Limit = 200
        .RequiredControl = msoTrue
    End With
End Sub
```



Resolution Property

Returns or sets a **String** that represents the resolution, in dots per inch (dpi), at which to print the specified publication. Default is dependent on the printer driver, but is usually "(default)". Read/write.

expression.**Resolution**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Valid values for the **Resolution** property depend on the printer driver being used. Printers have preset resolutions that cannot be customized. Values must be formatted in the following manner, including spacing:

"*HorizontalDotsPerInch* x *VerticalDotsPerInch*"

HorizontalDotsPerInch and *VerticalDotsPerInch* are numerical values, separated by one space, a lowercase x, and another space.

For example, to set the resolution of a printer to 600 horizontal dpi by 600 vertical dpi, a valid string would read "600 x 600".

The **Resolution** property also accepts the string "(default)" to specify the printer's default resolution setting. If the printer driver presents a language other than English, the **Resolution** property accepts the string that denotes the default setting in that language as well.

If the **Resolution** property is set to the default printer driver setting, using a **Get** statement returns the English string "(default)", regardless of whether the resolution was set to default using a non-English string.

This property corresponds to the **Resolution** control on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

The following example sets the resolution of the active publication at 300 dpi by 300 dpi. The example assumes that "300 x 300" is a valid string for the printer driver used.

```
ActiveDocument.AdvancedPrintOptions.Resolution = "300 x 300"
```

Result Property

Returns a **String** that represents the result of the specified field. Read-only.

expression.**Result**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example applies bold formatting to the first field in the selection. This example assumes that either text or a shape with text is selected in the active publication.

```
Sub GetFieldResults()
    If Selection.TextRange.Fields.Count > 0 Then
        MsgBox "The result of the first field is " & _
            Selection.TextRange.Fields(1).Result & "."
    End If
End Sub
```

ReturnDataLabel Property

Returns or sets a **String** that represents the text used by the Web page to label the specified Web object when the page is submitted. Read/write.

expression.**ReturnDataLabel**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web text box and specifies the label for the text in the text box when the page is submitted.

```
Sub LabelWebTextBoxControl()
    With ActiveDocument.Pages(1).Shapes _
            .AddWebControl(Type:=pbWebControlSingleLineTextBox, _
            Left:=100, Top:=100, Width:=300, Height:=15).WebTextBox
        .DefaultText = "Please enter your name here"
        .Limit = 70
        .RequiredControl = msoTrue
        .ReturnDataLabel = "Full_Name"
    End With
End Sub
```


Show All

RGB Property

Returns or sets an **MsoRGBType** that represents the red-green-blue (RGB) value of the specified color. Read/write.

expression.**RGB**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new shape to the first page of the active publication and sets the fill color to red.

```
Sub SetFill()
    ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShape5pointStar
        Left:=100, Top:=100, Width:=100, Height:=100).Fill.ForeColor
        .RGB = RGB(Red:=255, Green:=0, Blue:=0)
End Sub
```

This example returns the value of the foreground color of the first shape on the first page of the active document. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub ShowFillColor()
    MsgBox "The RGB fill value of this shape is " & _
        ActiveDocument.Pages(1).Shapes(1).Fill.ForeColor.RGB & "."
End Sub
```

RightIndent Property

Returns or sets a **Variant** that represents the right indent (in points) for the specified paragraphs. Read/write.

expression.**RightIndent**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the right indent for all paragraphs in the active document to one inch from the right margin. The **InchesToPoints** method is used to convert inches to points. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub SetRightIndent()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .TextRange.Paragraphs(1).ParagraphFormat _
        .RightIndent = InchesToPoints(1)
End Sub
```


Show All

RotatedChars Property

MsoTrue if characters in the specified WordArt are rotated 90 degrees relative to the WordArt's bounding shape. **MsoFalse** if characters in the specified WordArt retain their original orientation relative to the bounding shape. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**RotatedChars**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

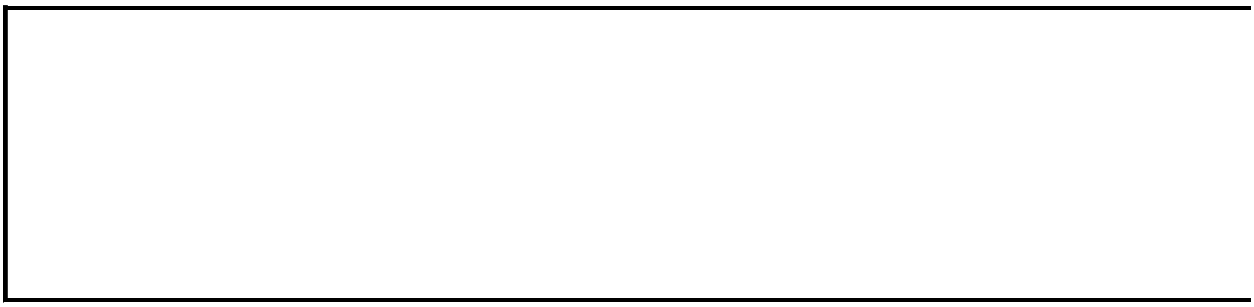
If the WordArt has horizontal text, setting the **RotatedChars** property to **True** rotates the characters 90 degrees counterclockwise. If the WordArt has vertical text, setting the **RotatedChars** property to **False** rotates the characters 90 degrees clockwise. Use the **ToggleVerticalText** method to switch between horizontal and vertical text flow.

The **Flip** method and **Rotation** property of the **Shape** object and the **RotatedChars** property and **ToggleVerticalText** method of the **TextEffectFormat** object all affect the character orientation and direction of text flow in a **Shape** object that represents WordArt. You may have to experiment to find out how to combine the effects of these properties and methods to get the result you want.

Example

This example adds WordArt that contains the text "Test" to the active publication and rotates the characters 90 degrees counterclockwise.

```
Sub CreateFormatWordArt()
    With ActiveDocument.Pages(1).Shapes _
            .AddTextEffect(PresetTextEffect:=msoTextEffect1, _
            Text:="Test", FontName:="Arial Black", FontSize:=36, _
            FontBold:=False, FontItalic:=False, Left:=10, Top:=10)
        .TextEffect.RotatedChars = msoTrue
    End With
End Sub
```



Rotation Property

Returns or sets a **Single** that represents the number of degrees the specified shape is rotated around the z-axis. A positive value indicates clockwise rotation; a negative value indicates counterclockwise rotation. Read/write.

expression.Rotation

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

To set the rotation of a three-dimensional shape around the x-axis or the y-axis, use the **RotationX** property or the **RotationY** property of the **ThreeDFormat** object.

Example

This example matches the rotation of all shapes on the first page of the active publication to the rotation of the first shape. This example assumes there are at least two shapes on the first page of the active publication.

```
Sub SetShapeRotation()
    Dim sngRotation As Single
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes
        sngRotation = .Item(1).Rotation
        For intCount = 1 To .Count
            .Item(intCount).Rotation = sngRotation
        Next intCount
    End With
End Sub
```

RotationX Property

Returns or sets the rotation of the extruded shape around the x-axis in degrees. Can be a value from – 90 through 90. A positive value indicates upward rotation; a negative value indicates downward rotation. Read/write **Single**.

expression.**RotationX**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

To set the rotation of the extruded shape around the y-axis, use the **RotationY** property of the **ThreeDFormat** object. To set the rotation of the extruded shape around the z-axis, use the **Rotation** property of the **Shape** object. To change the direction of the extrusion's sweep path without rotating the front face of the extrusion, use the **SetExtrusionDirection** method.

Example

This example adds three identical extruded ovals to the active document and sets their rotation around the x-axis to – 30, 0, and 30 degrees, respectively.

```
Sub SetRotationX()
    With ActiveDocument.Pages(1).Shapes
        With .AddShape(Type:=msoShapeOval, Left:=30, _
                Top:=60, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationX = -30
        End With
        With .AddShape(Type:=msoShapeOval, Left:=90, _
                Top:=60, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationX = 0
        End With
        With .AddShape(Type:=msoShapeOval, Left:=150, _
                Top:=60, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationX = 30
        End With
    End With
End Sub
```

RotationY Property

Returns or sets the rotation of the extruded shape around the y-axis, in degrees. Can be a value from – 90 through 90. A positive value indicates rotation to the left; a negative value indicates rotation to the right. Read/write **Single**.

expression.**RotationY**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

To set the rotation of the extruded shape around the x-axis, use the **RotationX** property of the **ThreeDFormat** object. To set the rotation of the extruded shape around the z-axis, use the **Rotation** property of the **Shape** object. To change the direction of the extrusion's sweep path without rotating the front face of the extrusion, use the **SetExtrusionDirection** method.

Example

This example adds three identical extruded ovals to the active document and sets their rotation around the y-axis to – 30, 0, and 30 degrees, respectively.

```
Sub SetRotationY()
    With ActiveDocument.Pages(1).Shapes
        With .AddShape(Type:=msoShapeOval, Left:=30, _
                Top:=120, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationY = -30
        End With
        With .AddShape(Type:=msoShapeOval, Left:=90, _
                Top:=120, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationY = 0
        End With
        With .AddShape(Type:=msoShapeOval, Left:=150, _
                Top:=120, Width:=50, Height:=25).ThreeD
            .Visible = True
            .RotationY = 30
        End With
    End With
End Sub
```

Row Property

Returns a **Long** that represents the row number containing the specified cell. Read-only.

expression.**Row**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enters the fill for all even-numbered rows and clears the fill for all odd-numbered rows in the specified table. This example assumes the specified shape is a table and not another type of shape.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell

    Set shpTable = ActiveDocument.Pages(1).Shapes _
        .AddTable(NumRows:=5, NumColumns:=5, Left:=100, _
        Top:=100, Width:=100, Height:=12)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            If celTable.Row Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
        Next celTable
    Next rowTable

End Sub
```

RowGutterWidth Property

Returns or sets the width of the row gutters that are used by the **LayoutGuides** object to aid in the process of laying out design elements. Read/write **Single**.

expression.**RowGutterWidth**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The default width of row gutters is 0.4 inches.

Example

The following example modifies the second master page of the active publication so that it has four rows and four columns, row gutter width of 0.75 inches, column gutter width of 0.5 inches, and center lines in the gutters. Any new pages added to the publication that use the second master page as a template will have these properties.

```
Dim theMasterPage As page
Dim theLayoutGuides As LayoutGuides

Set theMasterPage = ActiveDocument.MasterPages(2)
Set theLayoutGuides = theMasterPage.LayoutGuides

With theLayoutGuides
    .Rows = 4
    .Columns = 4
    .RowGutterWidth = Application.InchesToPoints(0.75)
    .ColumnGutterWidth = Application.InchesToPoints(0.5)
    .GutterCenterlines = True
End With
```


Show All

Rows Property

 Rows property as it applies to the **LayoutGuides** object.

Sets or returns a **Long** that represents the number of rows in a layout guide. Read/write.

expression.**Rows**

expression Required. An expression that returns one of the above objects.

 Rows property as it applies to the **Table** object.

Returns a **Rows** collection that represents all the table rows in a range, selection, or table.

expression.**Rows**

expression Required. An expression that returns one of the above objects.

Remarks

For information about returning a single member of a collection, see Returning an Object from a Collection.

Example

 As it applies to the **LayoutGuides** object.

This example sets the columns and rows for the layout guides.

```
Sub SetLayoutGuides()
    With ActiveDocument.LayoutGuides
        .Columns
        .Rows
    End With
End Sub
```

 As it applies to the **Table** object.

This example enters the fill for all even-numbered rows and clears the fill for all odd-numbered rows in the specified table. This example assumes the specified shape is a table and not another type of shape.

```
Sub FillCellsByRow()
    Dim shpTable As Shape
    Dim rowTable As Row
    Dim celTable As Cell

    Set shpTable = ActiveDocument.Pages(1).Shapes _
        .AddTable(NumRows:=5, NumColumns:=5, Left:=100, _
        Top:=100, Width:=100, Height:=12)
    For Each rowTable In shpTable.Table.Rows
        For Each celTable In rowTable.Cells
            If celTable.Row Mod 2 = 0 Then
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=180, Green:=180, Blue:=180)
            Else
                celTable.Fill.ForeColor.RGB = RGB _
                    (Red:=255, Green:=255, Blue:=255)
            End If
        Next celTable
    Next rowTable
End Sub
```

RulerGuides Property

Returns a **RulerGuides** collection that represents grid lines used to align objects on a page.

expression.**RulerGuides**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates horizontal and vertical ruler guides every half inch on the first page of the active publication.

```
Sub SetRulerGuides()
    Dim intCount As Integer
    Dim intPos As Integer
    With ActiveDocument.Pages(1).RulerGuides
        For intCount = 1 To 16
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeVertical
        Next intCount
        intPos = 0
        For intCount = 1 To 21
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeHorizontal
        Next intCount
    End With
End Sub
```

SaveAutoRecoverInfo Property

True if Publisher automatically saves publications for recovery if the application is unexpectedly shut down. Read/write **Boolean**.

expression.**SaveAutoRecoverInfo**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **SaveAutoRecoverInfoInterval** property to specify how often auto recovery saves occur.

Example

This example enables the global auto recovery option and sets the save interval to every five minutes.

```
Sub SetAutoRecoverInfo()
    With Options
        .SaveAutoRecoverInfo = True
        .SaveAutoRecoverInfoInterval = 5
    End With
End Sub
```

SaveAutoRecoverInfoInterval Property

Returns or sets a **Long** that represents the time interval in minutes for automatically saving a publication for recovery if the application is unexpectedly shut down. Read/write.

expression.**SaveAutoRecoverInfoInterval**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enables the global auto recovery option and sets the save interval to every five minutes.

```
Sub SetAutoRecoverInfo()
    With Options
        .SaveAutoRecoverInfo = True
        .SaveAutoRecoverInfoInterval = 5
    End With
End Sub
```

Saved Property

Returns **True** if no changes have been made to a publication since it was last saved. Read-only **Boolean**.

expression.**Saved**

expression Required. An expression that returns one of the objects in the Applies To list.

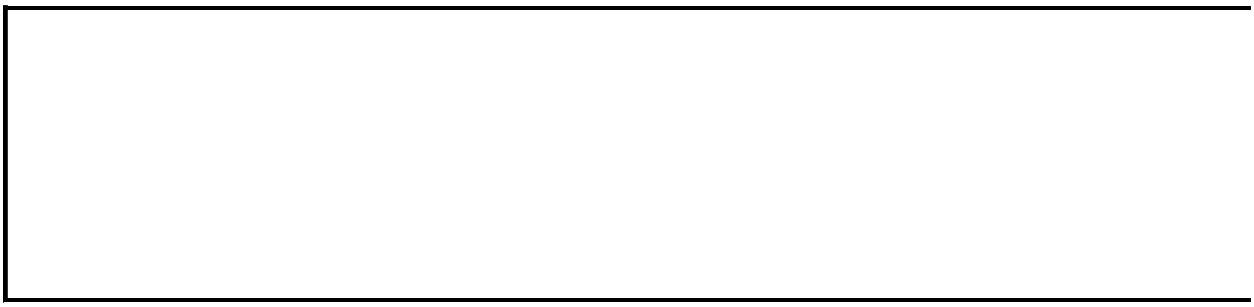
Remarks

If the **Saved** property of a modified publication returns **True**, the user won't be prompted to save changes when closing the publication, and all changes made to it since it was last saved will be lost.

Example

This example saves the active publication if it's been changed since the last time it was saved.

```
Sub Saver()

    With Application.ActiveDocument
        If Not .Saved And .Path <> "" Then .Save
    End With

End Sub
```




Show All

SaveFormat Property

Returns a **PbFileFormat** constant indicating the file format of the specified document. Read-only.

PbFileFormat can be one of these PbFileFormat constants.

pbFilePublication The file was saved with the current version of Publisher.

pbFilePublicationHTML The file was saved in an HTML format.

pbFilePublisher2000 The file was saved in a Publisher 2000 file format.

pbFilePublisher98 The file was saved in a Publisher 98 file format.

pbFileRTF The file was saved in Rich Text Format.

pbFileWebArchive The file was saved in the MHTML format that allows users to save a Web page and all its related files as a single file.

expression.**SaveFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

If the active publication is in the Publisher 2000 format, this example saves it in Rich Text Format (RTF).

```
Sub SaveAsRTF()

    If Application.ActiveDocument.SaveFormat = pbFilePublisher2000 T
        ActiveDocument.SaveAs "Flyer3", pbFileRTF
    End If

End Sub
```

Scaling Property

Returns or sets a **Variant** value used to scale the width of the characters in the text range as a percentage of the current font size. Read/write.

expression.**Scaling**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Valid range is 0.1 to 600.0 where the number represents the percentage of current font size. Indeterminate values are returned as –2.

Example

This example scales the width of the text in the second story by 200%. For this example to work, a second story with text must exist in the active document.

```
Sub ScaleUp()

    Application.ActiveDocument.Stories(2).TextRange.Font.Scaling = 2

End Sub
```


Show All

SchemeColor Property

Returns or sets a **PbSchemeColorIndex** constant that represents the specified color of the current color scheme. Read/write.

PbSchemeColorIndex can be one of these PbSchemeColorIndex constants.
pbSchemeColorAccent1
pbSchemeColorAccent2
pbSchemeColorAccent3
pbSchemeColorAccent4
pbSchemeColorAccent5
pbSchemeColorFollowedHyperlink
pbSchemeColorHyperlink
pbSchemeColorMain
pbSchemeColorNone

expression.**SchemeColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets the color of the text in shape one on page one of the active publication to accent color five in the current color scheme.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame _
    .TextRange.Font.Color.SchemeColor =
pbSchemeColorAccent5
```

ScratchArea Property

Returns a **ScratchArea** object for an a given document.

expression.**ScratchArea**

expression Required. An expression that returns one of the objects in the Applies To list.

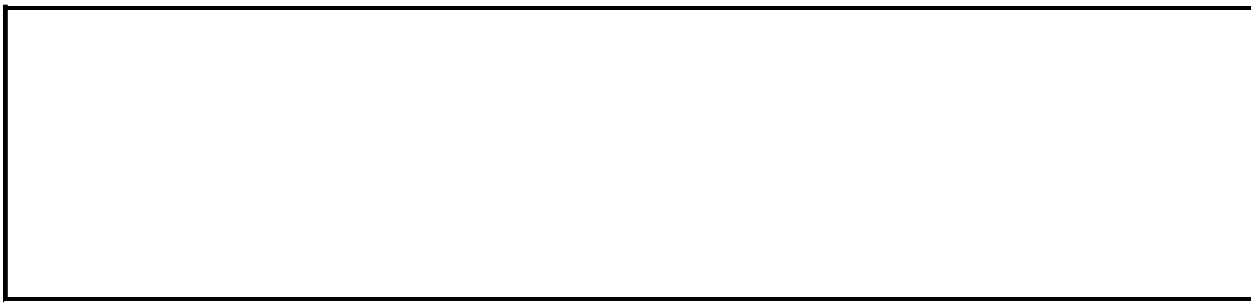
Remarks

The **ScratchArea** object is a collection of objects on the scratch page. The **ScratchArea** object is not in the **Pages** collection because it is fundamentally not a page; its only similarity to a page is that it can contain objects.

Example

This example sets the variable object as the first shape on the scratch area of the active document.

```
Sub ScratchPad()

    Dim saPage As ScratchArea
    Dim objFirst As Object

    saPage = Application.ActiveDocument.ScratchArea
    objFirst = saPage.Shapes(1)

End Sub
```



ScreenUpdating Property

Returns or sets a **Boolean** indicating whether Microsoft Publisher refreshes the screen display during run time; **True** to refresh the screen display. Read/write.

expression.**ScreenUpdating**

expression Required. An expression that returns one of the objects in the Applies To list.

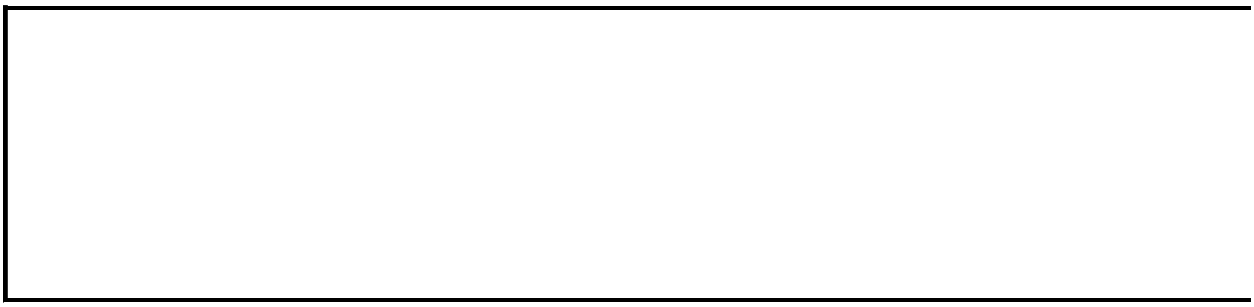
Remarks

Turning screen updating off during run time can speed execution of Visual Basic code. However, it is recommended to provide some indication of status so that the user is aware that the program is functioning correctly.

Example

The following example turns off screen updating at the beginning of a subroutine and turns it back on at the end of the subroutine.

```
Sub TurnOffScreenUpdating()
    ScreenUpdating = False

    ' Execute code here.

    ScreenUpdating = True
End Sub
```




Show All

Script Property

Returns a **PbFontScriptType** constant that represents the font script for a text range. Read-only.

PbFontScriptType can be one of these PbFontScriptType constants.

pbFontScriptArabic
pbFontScriptArmenian
pbFontScriptAsciiLatin
pbFontScriptAsciiSym
pbFontScriptBengali
pbFontScriptBopomofo
pbFontScriptBraille
pbFontScriptCanadianAbor
pbFontScriptCherokee
pbFontScriptCurrency
pbFontScriptCyrillic
pbFontScriptDefault
pbFontScriptDevanagari
pbFontScriptEthiopic
pbFontScriptEUDC
pbFontScriptGeorgian
pbFontScriptGreek
pbFontScriptGujarati
pbFontScriptGurmukhi
pbFontScriptHalfWidthKana
pbFontScriptHan
pbFontScriptHangul
pbFontScriptHanSurrogate
pbFontScriptHebrew
pbFontScriptKana
pbFontScriptKannada

pbFontScriptKhmer
pbFontScriptLao
pbFontScriptLatin
pbFontScriptMalayalam
pbFontScriptMixed
pbFontScriptMongolian
pbFontScriptMyanmar
pbFontScriptNonHanSurrogate
pbFontScriptOgham
pbFontScriptOriya
pbFontScriptRunic
pbFontScriptSinhala
pbFontScriptSyriac
pbFontScriptTamil
pbFontScriptTelugu
pbFontScriptThaana
pbFontScriptThai
pbFontScriptTibetan
pbFontScriptYi

expression.**Script**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays a message if the font script used in the specified text range is ASCII Latin. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub DisplayScriptType()
    If ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
            .Script = pbFontScriptAsciiLatin Then
        MsgBox "The font script you are using is ASCII Latin."
    End If
End Sub
```

Sections Property

Returns a **Sections** object representing a collection of **Section** objects in the specified document. Read-only **Sections**.

expression.**Sections**

expression Required. An expression that returns a **Document** object.

Example

This example sets an object variable to the **Sections** object of the active document and adds a new section starting at the second page of the publication. This example assumes that there are at least two pages in the publication.

```
Dim objSections As Sections
Set objSections = ActiveDocument.Sections
objSections.Add StartPageIndex:=2
```


Show All

SegmentType Property

Returns an **MsoSegmentType** constant that indicates whether the segment associated with the specified node is straight or curved. Read-only.

MsoSegmentType can be one of these MsoSegmentType constants.
msoSegmentCurve
msoSegmentLine

expression.**SegmentType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the specified node is a control point for a curved segment, this property returns **msoSegmentCurve**.

Use the **SetSegmentType** method to set the value of this property.

Example

This example changes all straight segments to curved segments in the first shape on the first page of the active publication. For this example to work, the specified shape must be a freeform drawing.

```
Sub ChangeSegmentTypes()
    Dim intNode As Integer
    With ActiveDocument.Pages(1).Shapes(1).Nodes
        intNode = 1
        Do While intNode <= .Count
            If .Item(intNode).SegmentType = msoSegmentLine Then
                .SetSegmentType Index:=intNode, _
                    SegmentType:=msoSegmentCurve
            End If
            intNode = intNode + 1
        Loop
    End With
End Sub
```


Show All

Selected Property

 Selected property as it applies to the **WebCheckBox** and **WebOptionButton** objects.

Returns or sets an **MsoTriState** constant that represents whether a Web check box or option button is selected. Read/write.

MsoTriState can be one of these MsoTriState constants.
msoCTrue
msoFalse
msoTriStateMixed
msoTriStateToggle
msoTrue

expression.**Selected**

expression Required. An expression that returns one of the above objects.

 Selected property as it applies to the **Cell** object.

True if a cell is selected. Read-only **Boolean**.

expression.**Selected**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **WebCheckBox** object.

This example adds a new Web check box to the first page of the active publication and then selects it.

```
Sub AddNewWebCheckBox()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlCheckBox, Left:=100, _
            Top:=100, Width:=100, Height:=12)
        .WebCheckBox.Selected = msoTrue
    End With
End Sub
```

As it applies to the **Cell** object.

This example determines if a cell in the specified table is selected and if it is, enters text into the cell.

```
Sub IsCellSelected()
    Dim cel As Cell
    With ActiveDocument.Pages(1).Shapes(1).Table
        For Each cel In .Cells
            If cel.Selected Then
                cel.TextRange.Text = "This cell is selected."
            End If
        Next cel
    End With
End Sub
```

Selection Property

Returns a **Selection** object that represents a selected range or the insertion point.

expression.**Selection**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests whether the current selection is text. If it is text, the selected text is then displayed in a message box.

```
Sub Selectable()

    If Selection.Type = pbSelectionText Then MsgBox Selection.TextRa

End Sub
```

SequenceCheck Property

True to check the sequence of independent characters for Asian text. Read/write **Boolean**.

expression.**SequenceCheck**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enables sequence checking, allowing the user to input a valid sequence of independent characters to form valid character cells in South Asian text.

```
Sub CheckSequence()
    Options.SequenceCheck = True
End Sub
```


Show All

Shadow Property

 Shadow property as it applies to the **Font** object.

MsoTrue if the specified font is formatted as shadowed. Read/write **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.
msoCTrue
msoFalse
msoTriStateMixed
msoTriStateToggle
msoTrue

expression.**Shadow**

expression Required. An expression that returns one of the above objects.

 Shadow property as it applies to the **Shape** and **ShapeRange** objects.

Returns a **ShadowFormat** object that represents the shadow formatting for the specified shape.

expression.**Shadow**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **Font** object.

This example applies shadow and bold formatting to the selection. This example assumes text is selected in the active publication.

```
Sub SetFontShadow()
    If Selection.Type = pbSelectionText Then
        With Selection.TextRange.Font
            .Shadow = msoTrue
            .Bold = msoTrue
        End With
    Else
        MsgBox "You need to select some text."
    End If
End Sub
```

As it applies to the **Shape** and **ShapeRange** objects.

This example adds an arrow with shadow formatting and fill color to the first page in the active document.

```
Sub SetShapeShadow()
    With ActiveDocument.Pages(1).Shapes.AddShape( _
            Type:=msoShapeRightArrow, Left:=72, _
            Top:=72, Width:=64, Height:=43)
        .Shadow.Type = msoShadow5
        .Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=255)
    End With
End Sub
```

Shape Property

Returns a **Shape** object that represents the shape associated with a hyperlink.

expression.**Shape**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a hyperlink to the first shape on the first page of the active publication and then vertically flips the shape. This example assumes there is at least one shape on the first page of the active publication.

```
Sub FormatHyperlinkShape()
    With ActiveDocument.Pages(1).Shapes(1).Hyperlink
        .Address = "http://www.tailspintoys.com/"
        .Shape.Flip FlipCmd:=msoFlipVertical
    End With
End Sub
```

ShapeRange Property

Returns a **ShapeRange** collection that represents all the **Shape** objects in the specified range or selection. The shape range can contain drawings, shapes, pictures, OLE objects, ActiveX controls, text objects, and callouts.

expression.**ShapeRange**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets the fill pattern for all the shapes in the selection. This example assumes one or more shapes are selected in the active publication.

```
Sub ChangeFillForShapeRange()
    Selection.ShapeRange.Fill.Patterned Pattern:=msoPattern20Percent
End Sub
```

The following example applies shadow and fill formatting to all the shapes in the selection. This example assumes one or more shapes are selected in the active publication.

```
Sub SetShadowForSelectedShapes()
    With Selection.ShapeRange
        .Shadow.Type = msoShadow6
        .Fill.Patterned Pattern:=msoPatternDottedDiamond
    End With
End Sub
```

Shapes Property

Returns a **Shapes** collection that represents all the **Shape** objects in the specified publication. This collection can contain drawings, shapes, pictures, OLE objects, ActiveX controls, text objects, and callouts.

expression.**Shapes**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

For information about returning a single member of a collection, see Returning an Object from a Collection.

Example

This example adds a rectangle to the first page in the active publication.

```
Sub AddNewRectangle()
    ActiveDocument.Pages(1).Shapes.AddShape Type:=msoShapeRectangle,
        Left:=5, Top:=25, Width:=100, Height:=50
End Sub
```

This example sets the fill texture for all the shapes in the active publication. This example assumes there is at least one shape in the active publication.

```
Sub SetNewTextureForAllShapes()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        shp.Fill.PresetTextured PresetTexture:=msoTextureOak
    Next shp
End Sub
```

This example adds a shadow to the first shape in the active publication. This example assumes there is at least one shape in the active publication.

```
Sub SetShadowForFirstShape()
    ActiveDocument.Pages(1).Shapes(1).Shadow.Type = msoShadow6
End Sub
```

This example displays a count of all shapes on the first page of the active publication. This example assumes there is at least one shape in the active publication.

```
Sub CountShapesOnFirstPage()
    MsgBox "You have " & ActiveDocument.Pages(1) _
        .Shapes.Count & " shapes on the first page."
End Sub
```

ShowBasicColors Property

Returns or sets a **Boolean** indicating whether Microsoft Publisher shows basic colors in the color palette; **True** to show basic colors in the palette. Read/write.

expression.**ShowBasicColors**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets Publisher to not show basic colors in the color palette.

```
Options.ShowBasicColors = False
```

ShowHeaderFooterOnFirstPage Property

True if the header and footer of the specified section will be visible. Read/write **Boolean**.

expression.**ShowHeaderFooterOnFirstPage**()

expression Required. An expression that returns a **Section** object.

Example

The fiollowing example adds a new section starting on the second page of the activedocument, adds header and footer text to the master page, and then sets the **ShowHeaderFooterOnFirstPage** property to **True**.

```
Dim objSection As Section
Set objSection = ActiveDocument.Sections.Add(StartPageIndex:=2)
With ActiveDocument.Pages(2).Master
    .Header.TextRange.Text = "Page " & .PageNumber & " header."
    .Footer.TextRange.Text = "Page " & .PageNumber & " footer."
End With
objSection.ShowHeaderFooterOnFirstPage = True
```

ShowOnlyWebFonts Property

Returns or sets a **Boolean** value that specifies whether only Web-safe fonts and font schemes should be used when the Web site is viewed in a browser. If **True**, only Web-safe fonts and font schemes are used. If **False**, display is not limited to Web-safe fonts and font schemes. The default value is **False**. Read/write.

expression.**ShowOnlyWebFonts**

expression Required. An expression that returns a **WebOptions** object.

Remarks

This property applies to Latin-based fonts only.

Example

The following example specifies that only Web-safe fonts and font schemes should be used when the Web site is viewed in a browser.

```
Dim theWO As WebOptions

Set theWO = Application.WebOptions

With theWO
    .ShowOnlyWebFonts = True
End With
```

ShowScreenTipsOnObjects Property

True for Microsoft Publisher to display ScreenTips when the mouse cursor hovers over a text box, shape or other object. Read/write **Boolean**.

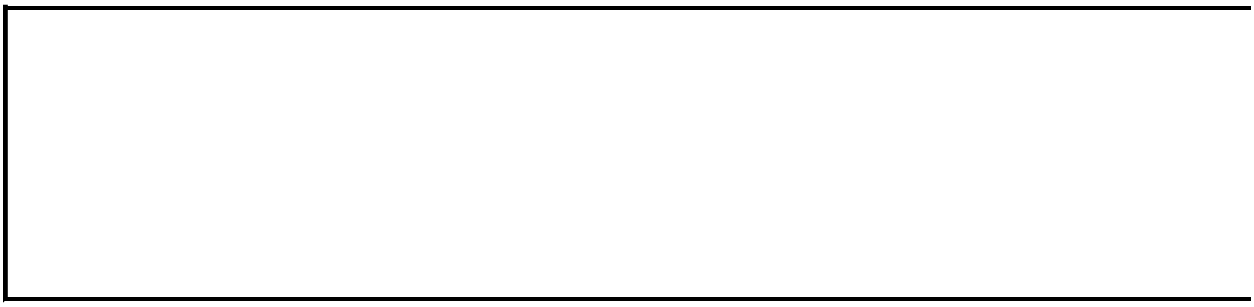
expression.**ShowScreenTipsOnObjects**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example disables displaying ScreenTips on objects.

```
Sub DisableScreenTips()
    Options.ShowScreenTipsOnObjects = False
End Sub
```



ShowSelected Property

True if the selected button is highlighted for the specified **WebNavigationBarSet** object. Read/write **Boolean**.

expression.**ShowSelected**

expression Required. An expression that returns a **WebNavigationBarSet** object.

Example

The following example adds a new Web navigation bar to the active document, adds it to every page, and then sets the **ShowSelected** property to **False** so that the selected button will not be highlighted in the navigation bar.

```
Dim objWebNav As WebNavigationBarSet
Set objWebNav = ActiveDocument.WebNavigationBarSets.AddSet(Name:="ne
With objWebNav
    .AddToEveryPage Left:=10, Top:=10
    .ButtonStyle = pbnbButtonStyleSmall
    .ShowSelected = False
End With
```

ShowTipPages Property

True for Microsoft Publisher to display tippages in balloons. Read/write **Boolean**.

expression.**ShowTipPages**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example disables displaying tippages in balloons.

```
Sub DontShowTipPages()
    Options.ShowTipPages = False
End Sub
```


Show All

Side Property

Returns or sets a **PbWrapSideType** constant that indicates whether text should wrap around a shape. Read/write.

PbWrapSideType can be one of these PbWrapSideType constants.
pbWrapSideBoth
pbWrapSideLarger
pbWrapSideLeft
pbWrapSideMixed
pbWrapSideNeither
pbWrapSideRight

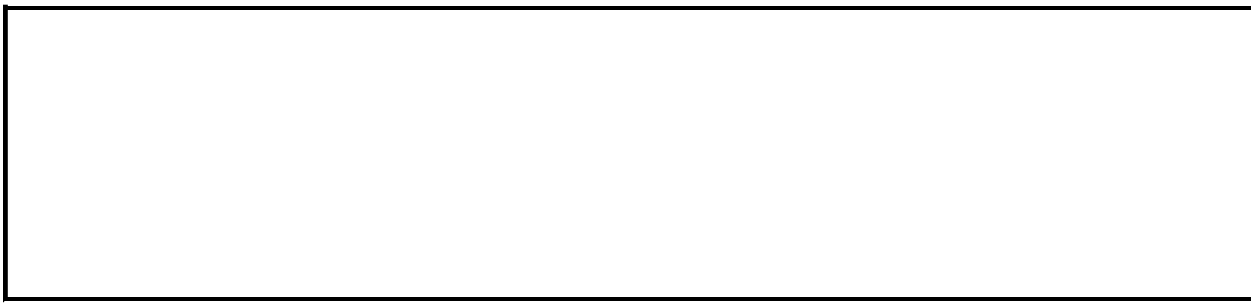
expression.**Side**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds an oval to the first page of the active publication and specifies that text wrap around both the left and right sides of the oval.

```
Sub SetTextWrapFormatProperties()
    With ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShapeOval,
             Left:=36, Top:=36, Width:=100, Height:=35)
        With .TextWrap
            .Type = pbWrapTypeSquare
            .Side = pbWrapSideBoth
        End With
    End With
End Sub
```




Show All

Size Property

 Size property as it applies to the **DropCap** object.

Returns or sets a **Long** that represents the number of lines high to format a dropped capital letter. Read/write.

expression.**Size**

expression Required. An expression that returns one of the above objects.

 Size property as it applies to the **Font** object.

Returns or sets a **Variant** that represents the size of the characters in the text range in points. Read/write.

expression.**Size**

expression Required. An expression that returns one of the above objects.

Remarks

The valid range for the **Size** property is 0.5 points to 999.5 points. The **Size** property returns –2 if the size of characters is indeterminate.

Example

 As it applies to the **DropCap** object.

This example formats a drop cap for the specified text range that is five lines high.

```
Sub RaisedDropCap()
    Dim intCount As Integer
    With ActiveDocument.Pages(1).Shapes _
            .AddTextbox(Orientation:=pbTextOrientationHorizontal, _
            Left:=100, Top:=100, Width:=100, Height:=100) _
            .TextFrame.TextRange
        For intCount = 1 To 10
            .InsertAfter NewText:="This is a test. "
        Next intCount
        With .DropCap
            .Size = 5
            .LinesUp = 2
        End With
    End With
End Sub
```

 As it applies to the **Font** object.

This example inserts text and then sets the font size of the seventh word of the inserted text to 20 points.

```
Sub IncreaseFontSizeOfSelection()
    With Selection.TextRange
        .InsertBefore vbLf & "This is a demonstration of font size."
        .Words(7).Font.Size = 20
    End With
End Sub
```

SizeBi Property

Returns or sets a **Variant** value representing the size, in points, of the **Font** object for text in a right-to-left language. Valid range is 0.5 points to 999.5 points. Read/write.

expression.**SizeBi**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example tests the text in the second story. If it is in a right-to-left language, larger than 12 point, and italicized, the text is set to bold.

```
Sub SizeBiIfBig()

    Dim fntSize As Font

    Set fntSize = Application.ActiveDocument.Stories(2).TextRange.Fo
    With fntSize
        If .SizeBi > 12 And .ItalicBi = msoTrue Then
            .BoldBi = msoTrue
        Else
            MsgBox "The font size is 12 points or less " & _
            ", not bold, or this is not in a right-to-left language.
        End If
    End With

End Sub
```


Show All

SmallCaps Property

Returns or sets an **MsoTriState** constant indicating whether the specified text is formatted as small caps. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse None of the characters in the range are formatted as small caps.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified text range.

msoTriStateToggle Set value which toggles between **msoTrue** to **msoFalse**.

msoTrue All of the characters in the range are formatted as small caps.

expression.**SmallCaps**

expression Required. An expression that returns one of the objects in the Applies To list.

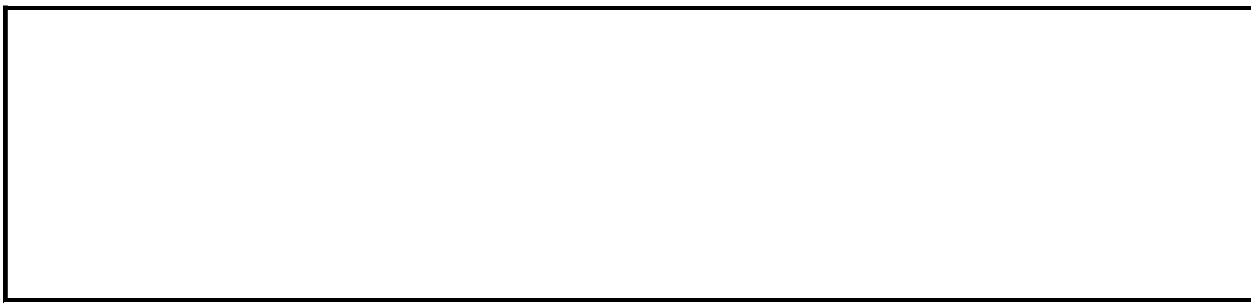
Remarks

Setting **SmallCaps** to **msoTrue** will remove all caps formatting from the text range.

Example

This example tests the text in the second story and if it has mixed small caps formatting, it formats all the text as small caps.

```
Sub SmallCap()

    Dim fntSC As Font

    Set fntSC = Application.ActiveDocument.Stories(2).TextRange.Font
    With fntSC
        If .SmallCaps = msoTriStateMixed Then
            .SmallCaps = msoTrue
        Else
            MsgBox "Mixed small caps are not in this story."
        End If
    End With

End Sub
```



SnapToGuides Property

True for Publisher to use the guides to align objects on a page in a publication. Read/write **Boolean**.

expression.**SnapToGuides**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds horizontal and vertical ruler guides every half inch on the first page and then sets the options to align objects on the page to the guides.

```
Sub SetSnapOptions()
    Dim intCount As Integer
    Dim intPos As Integer
    With ActiveDocument.Pages(1).RulerGuides
        For intCount = 1 To 16
            intPos = intPos + 36
            .Add Position:=intPos, Type:=pbRulerGuideTypeVertical
        Next
        intPos = 0
        For intCount = 1 To 21
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeHorizontal
        Next
    End With
    With Application
        .SnapToGuides = True
        .SnapToObjects = True
    End With
End Sub
```

SnapToObjects Property

True for Microsoft Publisher to use objects on a page to line up other objects. Read/write **Boolean**.

expression.**SnapToObjects**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds horizontal and vertical ruler guides every half inch on the first page and sets the options to align objects on the page to the guides.

```
Sub SetSnapOptions()
    Dim intCount As Integer
    Dim intPos As Integer
    With ActiveDocument.Pages(1).RulerGuides
        For intCount = 1 To 16
            intPos = intPos + 36
            .Add Position:=intPos, Type:=pbRulerGuideTypeVertical
        Next
        intPos = 0
        For intCount = 1 To 21
            intPos = intPos + 36
             .Add Position:=intPos, Type:=pbRulerGuideTypeHorizontal
        Next
    End With
    With Application
        .SnapToGuides = True
        .SnapToObjects = True
    End With
End Sub
```

SourceFullName Property

Returns a **String** that represents the path and name of the source file for the specified linked OLE object, picture, or field. Read-only.

expression.**SourceFullName**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example displays the path and file name of the source file for all embedded OLE shapes on the first page of the active publication.

```
Sub DisplaySourceName()
    Dim shp As Shape
    For Each shp In ActiveDocument.Pages(1).Shapes
        If shp.Type = pbEmbeddedOLEObject Then
            With shp.LinkFormat
                MsgBox .SourceFullName
            End With
        End If
    Next
End Sub
```

SpaceAfter Property

Returns or sets a **Variant** that represents the amount of spacing (in points) after one or more paragraphs. Read/write.

expression.**SpaceAfter**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the spacing before and after the third paragraph in the first shape on the first page of the active publication to 6 points.

```
Sub SetSpacingBeforeAfterParagraph()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.Paragraphs(3).ParagraphFormat
        .SpaceBefore = 6
        .SpaceAfter = 6
    End With
End Sub
```

This example sets spacing before and after all paragraphs in the first shape on the first page of the active publication to 6 points.

```
Sub SetSpacingBeforeAfterAllParagraph()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.ParagraphFormat
        .SpaceBefore = 12
        .SpaceAfter = 6
    End With
End Sub
```

SpaceBefore Property

Returns or sets a **Variant** that represents the amount of spacing (in points) before one or more paragraphs. Read/write.

expression.**SpaceBefore**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the spacing before and after the third paragraph in the first shape on the first page of the active publication to 6 points. This example assumes there is at least one shape on the first page of the active publication.

```
Sub SetSpacingBeforeAfterParagraph()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.Paragraphs(3).ParagraphFormat
        .SpaceBefore = 6
        .SpaceAfter = 6
    End With
End Sub
```

This example sets spacing before and after all paragraphs in the first shape on the first page of the active publication to 6 points. This example assumes there is at least one shape on the first page of the active publication.

```
Sub SetSpacingBeforeAfterAllParagraph()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame _
            .TextRange.ParagraphFormat
        .SpaceBefore = 12
        .SpaceAfter = 6
    End With
End Sub
```

Span Property

Returns or sets a **Long** that represents the number of letters included in the specified dropped capital letter. Read/write.

expression.**Span**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a custom dropped capital letter that is five lines high, spans the first three characters of the paragraphs in the text range, and is raised one line above the first line.

```
Sub SetDropCapSpan()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.DropCap
        .Size = 5
        .Span = 3
        .LinesUp = 1
    End With
End Sub
```

Start Property

Returns or sets a **Long** that represents the starting character position of a text range. Read/write.

expression.**Start**

expression Required. An expression that returns one of the objects in the Applies To list.

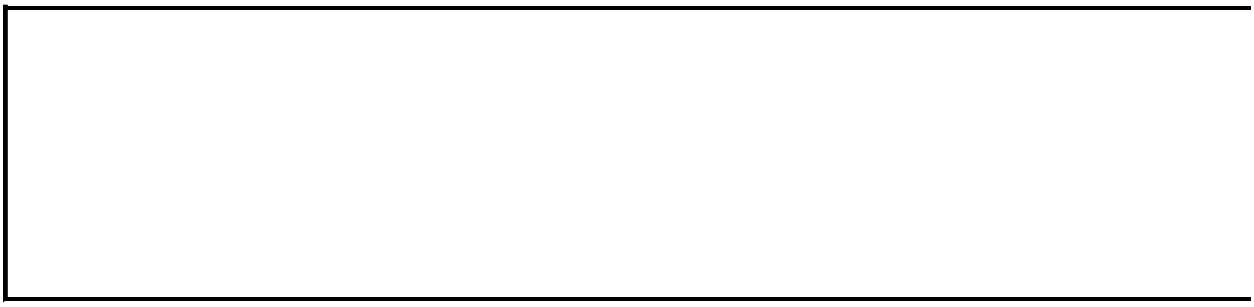
Remarks

If this property is set to a value larger than that of the **End** property, the **End** property is set to the same value as that of **Start** property.

Example

This example bolds the first fifteen characters of the selected text range. This example assumes that text is selected in the active publication.

```
Sub SetSelectionRange()
    With Selection
        With .TextRange
            .Start = 0
            .End = 15
            .Font.Bold = msoTrue
        End With
    End With
End Sub
```




Show All

StartInNextTextBox Property

Returns or sets an **MSOTriState** constant that represents whether to always start the selected paragraph in the next linked text box. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**StartInNextTextBox**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If text is added to the previous text box, causing text to overflow into the text box containing the specified text, the specified text (and any text following it) is moved to the top of the next available text box. If no linked text box is available, the specified text (and any text following it) is placed into the text overflow buffer. It will remain in the buffer until either another linked text box is added to the publication, or the **StartInNextTextBox** property is changed.

This property corresponds to the **Start in next text box** control on the **Paragraph** dialog box.

StartPageIndex Property

Returns the page number of the page that the specified **Section** object begins on. Read/write **Long**.

expression.**StartPageIndex**

expression Required. An expression that returns a **Section** object.

Example

The following example adds two pages to the active document, then sets the start page index of the first section to 3. This effectively adds a new section starting on the third page of the publication.

```
ActiveDocument.Pages.Add Count:=2, After:=1
ActiveDocument.Sections(1).StartPageIndex = 3
```

Stories Property

Returns a **Stories** collection containing all stories in the publication.

expression.**Stories**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example assigns the first story in the **Stories** collection to a variable.

```
Sub FirstStory()

    Dim stFirst As Story

    stFirst = Application.ActiveDocument.Stories(1)

End Sub
```

Story Property

Returns a **Story** object that represents the story properties in a text range.

expression.**Story**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example returns the story in the selected text range and, if it is in a text frame, inserts text into the text range.

```
Sub AddTextToStory()
    With Selection.TextRange.Story
        If .HasTextFrame Then
            .TextRange.InsertAfter NewText:=vbLf & "This is a test."
        End If
    End With
End Sub
```

StretchPictures Property

True to stretch the picture art making up the specified BorderArt to fit the shape to which it is applied. Read/write **Boolean**.

expression.**StretchPictures**()

expression Required. An expression that returns a **BorderArtFormat** object.

Remarks

Returns "Permission Denied" if BorderArt has not been applied to the specified object.

Corresponds to the **Don't stretch pictures** and **Stretch pictures to fit** controls on the **BorderArt** dialog box.

Example

The following example tests for the existence of BorderArt on each shape for each page of the active document. If BorderArt exists, it is set so that it can be stretched.

```
Sub StretchBorderArt()
        Dim anyPage As Page
        Dim anyShape As Shape

        For Each anyPage in ActiveDocument.Pages
                For Each anyShape in anyPage.Shapes
                        With anyShape.BorderArt
                                If .Exists = True Then
                                        .StretchPictures = True
                                End If
                        End With
                Next anyShape
        Next anyPage
End Sub
```


Show All

Style Property

Returns or sets an **MsoLineStyle** constant that represents the style of line to apply to a shape or border. Read/write.

MsoLineStyle can be one of these MsoLineStyle constants.
msoLineSingle
msoLineStyleMixed
msoLineThickBetweenThin
msoLineThickThin
msoLineThinThick
msoLineThinThin

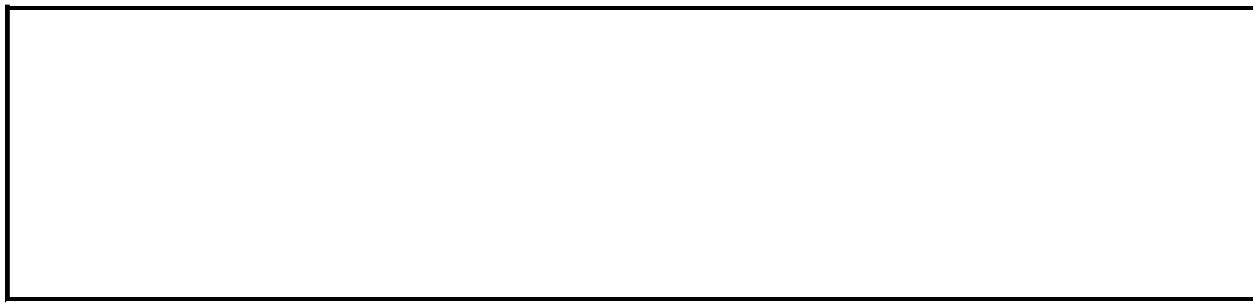
expression.**Style**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a new shape and sets the line properties for the shape.

```
Sub SetLineStyle()
    With ActiveDocument.Pages(1).Shapes.AddShape(msoShapeRectangle,
            Left:=72, Top:=140, Width:=200, Height:=100)
        .Rotation = 120
        With .Line
            .Weight = 5
            .DashStyle = msoLineDashDotDot
            .Style = msoLineThickBetweenThin
        End With
    End With
End Sub
```




Show All

SubScript Property

Returns or sets an **MsoTriState** constant indicating whether characters are formatted as subscript in the specified text range. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse No characters in the range are formatted as subscript.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse**.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All characters in the range are formatted as subscript.

expression.**SubScript**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Setting the **SubScript** property to **msoTrue** will remove superscript formatting from the text range.

Example

This example tests the text in the second story, and if it has mixed subscripting, it formats all the text as subscript.

```
Sub SubScript()

    Dim fntSS As Font

    Set fntSS = Application.ActiveDocument.Stories(2).TextRange.Font
    With fntSS
        If .SubScript = msoTriStateMixed Then
            .SubScript = msoTrue
        Else
            MsgBox "Mixed subscript not in this story."
        End If
    End With

End Sub
```


Show All

SuperScript Property

Returns or sets an **MsoTriState** constant indicating whether characters are formatted as superscript in the specified text range. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Does not apply to this property.

msoFalse No characters in the range are formatted as superscript.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse**.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue All characters in the range are formatted as superscript.

expression.**SuperScript**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Setting **SuperScript** property to **msoTrue** will remove subscript formatting from the text range.

Example

This example tests the text in the second story, and if it has mixed superscripting, it formats all the text as superscript.

```
Sub SuperScript()

    Dim fntSuper As Font

    Set fntSuper = Application.ActiveDocument.Stories(2).TextRange.F
    With fntSuper
        If .SuperScript = msoTriStateMixed Then
            .SuperScript = msoTrue
        Else
            MsgBox "Mixed superscript not in this story."
        End If
    End With

End Sub
```

SuppressBlankLines Property

True to suppress blank lines when mail merge fields in a mail merge main document are empty. Read/write **Boolean**.

expression.**SuppressBlankLines**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example suppresses blank lines in the active publication when mail merge fields are blank. This example assumes that a mail merge data source is attached to the active publication.

```
Sub SuppressBlankLines()
    ActiveDocument.MailMerge.SuppressBlankLines = True
End Sub
```

Table Property

Returns a **Table** object that represents a table in Microsoft Publisher.

expression.**Table**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example adds a 5x5 table on the first page of the active publication, and then selects the first column of the new table.

```
Sub NewTable()
    With ActiveDocument.Pages(1).Shapes.AddTable(NumRows:=5, _
        NumColumns:=5, Left:=72, Top:=300, Width:=400, Height:=100)
        .Table.Columns(3).Cells(3).Fill.ForeColor.RGB = RGB _
        (Red:=255, Green:=0, Blue:=0)
    End With
End Sub
```

The following example selects the specified table in the active publication. This example assumes that there is at least one shape on the first page of the active publication.

```
Sub SelectTable()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Type = pbTable Then
            .Table.Rows(3).Cells(3).Fill.ForeColor _
                .RGB = RGB(Red:=150, Green:=150, Blue:=150)
        End If
    End With
End Sub
```

TableCellRange Property

Returns a **CellRange** object that represents the cells in a table selection.

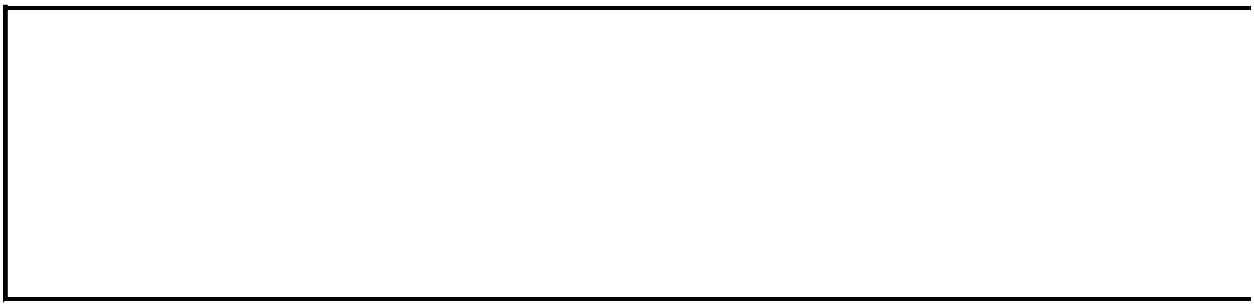
expression.**TableCellRange**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example fills the table cells in a selection.

```
Sub FillTableCellRange()
    Dim intCount As Integer
    With Selection
        If .Type = pbSelectionTableCells Then
            With .TableCellRange
                For intCount = 1 To .Count
                    .Item(intCount).Fill.ForeColor.RGB = RGB _
                        (Red:=0, Green:=255, Blue:=255)
                Next intCount
            End With
        End If
    End With
End Sub
```




Show All

TableDirection Property

Returns or sets a **PbTableDirectionType** constant that represents whether text in a table is read from left to right or from right to left. Read/write.

PbTableDirectionType can be one of these PbTableDirectionType constants.
pbTableDirectionLeftToRight
pbTableDirectionRightToLeft

expression.**TableDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example enters a bold number into each cell in the specified table, and then sets the direction of the table so that the cells number from right to left. For this example to work, the specified shape must be a table.

```
Sub CountCellsByColumn()
    Dim tblTable As Table
    Dim rowTable As row
    Dim celTable As Cell
    Dim intCount As Integer

    Set tblTable = ActiveDocument.Pages(1).Shapes(1).Table

    'Loops through each row in the table
    For Each rowTable In tblTable.Rows

        'Loops through each cell in the row
        For Each celTable In rowTable.Cells
            With celTable.TextRange
                intCount = intCount + 1
                .Text = intCount
                .ParagraphFormat.Alignment = _
                    pbParagraphAlignmentCenter
                .Font.Bold = msoTrue
            End With
        Next celTable
    Next rowTable
    tblTable.TableDirection = pbTableDirectionRightToLeft
End Sub
```

TableName Property

Returns a **String** that represents the name of the table within the data source file that contains the mail merge records. The returned value may be blank if the table name is unknown or not applicable to the current data source. Read-only.

expression.**TableName**

expression Required. An expression that returns a **MailMergeDataSource** object.

Example

This example displays a message with the name of the mail merge data source table name.

```
Sub EmployeeTable()
    With ActiveDocument.MailMerge.DataSource
        Select Case .TableName
            Case "Employees"
                MsgBox "This is an Employee mail merge publication."
            Case "Customers"
                MsgBox "This is a Customers mail merge publication."
            Case "Suppliers"
                MsgBox "This is a Suppliers mail merge publication."
            Case "Shippers"
                MsgBox "This is a Shippers mail merge publication."
            Case Else
                MsgBox "This is a " & .TableName & " mail merge publ
        End Select
    End With
End Sub
```

Tabs Property

Returns a **TabStops** object representing the custom and default tabs for a paragraph or group of paragraphs.

expression.**Tabs**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example adds two tab stops to the selected paragraphs. The first tab stop is a left-aligned tab with a dotted tab leader positioned at 1 inch (72 points). The second tab stop is centered and is positioned at 2 inches.

```
Dim tabsAll As TabStops

Set tabsAll = Selection.TextRange.ParagraphFormat.Tabs

With tabsAll
    .Add Position:=InchesToPoints(1), _
        Leader:=pbTabLeaderDot, Alignment:=pbTabAlignmentLeading
    .Add Position:=InchesToPoints(2), _
        Leader:=pbTabLeaderNone, Alignment:=pbTabAlignmentCenter
End With
```

Tags Property

Returns a **Tags** collection representing tags or custom properties applied to a shape, shape range, page, or publication.

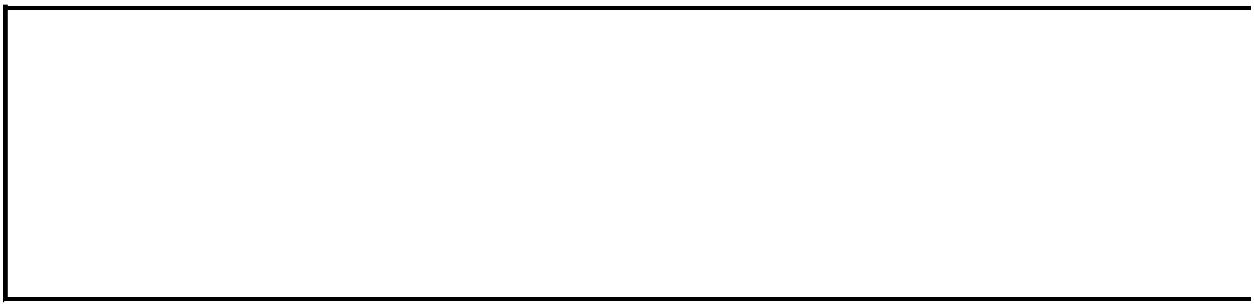
expression.**Tags**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a tag to each oval shape on the first page of the active publication.

```
Dim shp As Shape
Dim tagsAll As Tags
Dim tagLoop As Tag
Dim blnTag As Boolean

With ActiveDocument.Pages(1)
    For Each shp In .Shapes
        If shp.AutoShapeType = msoShapeOval Then
        Set tagsAll = shp.Tags
            blnTag = False

            For Each tagLoop In tagsAll
                If tagLoop.Name = "Shape" Then
                    blnTag = True
                    Exit For
                End If
            Next tagLoop

            If blnTag = False Then
                tagsAll.Add Name:="Shape", Value:="Oval"
            End If
        End If
    Next shp
End With
```




Show All

TargetType Property

Returns a **PbHlinkTargetType** constant that represents the type of hyperlink. Read-only.

PbHlinkTargetType can be one of these PbHlinkTargetType constants.
pbHlinkTargetTypeEmail
pbHlinkTargetTypeFirstPage
pbHlinkTargetTypeLastPage
pbHlinkTargetTypeNextPage
pbHlinkTargetTypeNone
pbHlinkTargetTypePageID
pbHlinkTargetTypePreviousPage
pbHlinkTargetTypeURL

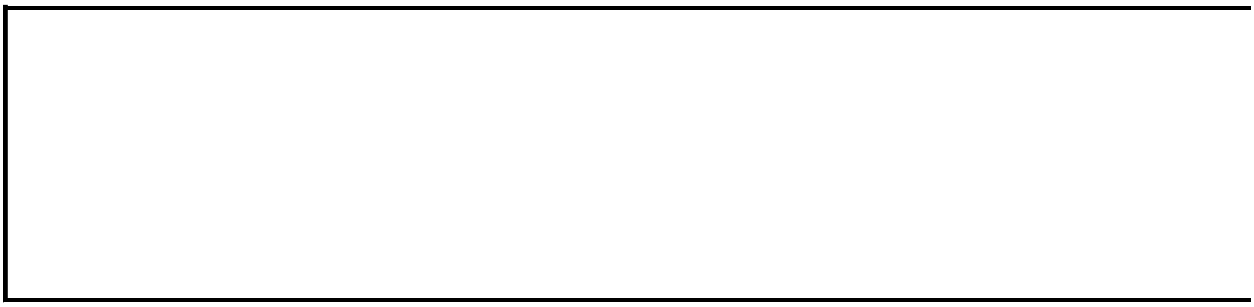
expression.**TargetType**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example verifies that the specified hyperlink is a URL and if it is, sets the hyperlink display text and address. This example assumes there is at least one shape on the first page of the active publication.

```
Sub SetHyperlinkTextToDisplay()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.Hyperlinks.Item(1)
        If .TargetType = pbHlinkTargetTypeURL Then
            .TextToDisplay = "Tailspin Toys Web Site"
            .Address = "http://www.tailspintoys.com/"
        End If
    End With
End Sub
```



TemplateFolderPath Property

Returns a **String** that represents the location where Publisher templates are stored. Read-only.

expression.**TemplateFolderPath**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new publication and edits the master page to contain a page number in a star in the upper left corner of the page; then it saves the new publication to the template folder location so it can be used as a template.

```
Sub CreateNewPubTemplate()
    Dim AppPub As Application
    Dim DocPub As Document
    Dim strFolder As String

    Set AppPub = New Publisher.Application
    Set DocPub = AppPub.NewDocument
    AppPub.ActiveWindow.Visible = True
    strFolder = AppPub.TemplateFolderPath

    With DocPub
        With .MasterPages(1).Shapes.AddShape _
                (Type:=msoShape5pointStar, Left:=36, _
                Top:=36, Width:=50, Height:=50)
            .Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)
            With .TextFrame.TextRange
                .InsertPageNumber
                .ParagraphFormat.Alignment = pbParagraphAlignmentCen
                With .Font
                    .Bold = msoTrue
                    .Color.RGB = RGB(Red:=255, Green:=255, Blue:=255
                    .Size = 12
                End With
            End With
        End With
        .SaveAs FileName:=strFolder & "\NewPubTemplt.pub"
    End With
End Sub
```


Show All

Text Property

 Text property as it applies to the **TextEffectFormat** and **TextRange** objects.

Returns or sets a **String** that represents the text in a text range or WordArt shape. Read/write.

expression.**Text**

expression Required. An expression that returns one of the above objects.

 Text property as it applies to the **PhoneticGuide** object.

Returns a **String** that represents the contents of phonetic text. Read-only.

expression.**Text**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **TextRange** object.

The following example adds a rectangle to the active publication and adds text to it.

```
Sub AddTextToShape()
    With ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShapeRecta
        Left:=72, Top:=72, Width:=250, Height:=140)
        .TextFrame.TextRange.Text = "Here is some test text"
    End With
End Sub
```

As it applies to the **TextEffectFormat** object.

The following example changes the text and sets the font name and formatting properties for shape one on the first page of the active publication. For this example to work, shape one must be a WordArt object.

```
Sub FormatWordArt()
    With ActiveDocument.Pages(1).Shapes(1).TextEffect
        .Text = "This is a test."
        .FontName = "Courier New"
        .FontBold = True
        .FontItalic = True
    End With
End Sub
```


Show All

TextDirection Property

Returns or sets a **PbTextDirection** constant indicating the direction in which text flows in the specified paragraph. Read/write.

PbTextDirection can be one of these PbTextDirection constants.

pbTextDirectionLeftToRight Text flows from left to right.

pbTextDirectionMixed Return value indicating a range containing some left-to-right text and some right-to-left text.

pbTextDirectionRightToLeft Text flows from right to left.

expression.**TextDirection**

expression Required. An expression that returns one of the objects in the Applies To list.

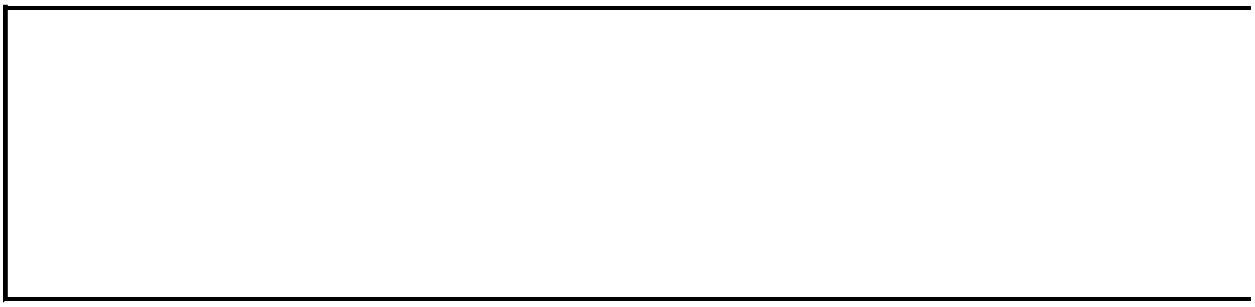
Remarks

This property is meant to be used in conjunction with documents that have text in both left-to-right and right-to-left languages. Setting the property to a value that is not in accordance with the text direction dictated by the language in use may have unpredictable results.

Example

The following example changes the text direction of the first shape on page one so that it flows from right-to-left.

```
ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange _
    .ParagraphFormat.TextDirection = pbTextDirectionRightToLeft
```



TextEffect Property

Returns a **TextEffectFormat** object that represents the text formatting properties of a WordArt object.

expression.**TextEffect**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example adds a WordArt object to the active publication and formats and inserts additional into it.

```
Sub AddFormatNewWordArt()
    With ActiveDocument.Pages(1).Shapes.AddTextEffect( _
            PresetTextEffect:=msoTextEffect1, Text:="Test", _
            FontName:="Snap ITC", FontSize:=30, FontBold:=msoTrue, _
            FontItalic:=msoFalse, Left:=150, Top:=130)
        .Rotation = 90
        With .TextEffect
            .RotatedChars = msoTrue
            .Text = "This is a " & .Text
        End With
        .Width = 250
    End With
End Sub
```

TextFrame Property

Returns a **TextFrame** object that represents the text in a shape as well as the properties that control the margins and orientation of the text.

expression.**TextFrame**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example adds text to the text frame of shape one in the active publication, and then formats the new text. This example assumes there is at least one shape on the first page of the active publication.

```
Sub AddTextToTextFrame()
    With ActiveDocument.Pages(1).Shapes(1).TextFrame.TextRange
        .Text = "My Text"
        With .Font
            .Bold = msoTrue
            .Size = 25
            .Name = "Arial"
        End With
    End With
End Sub
```

TextRange Property

Returns a **TextRange** object that represents the text that's attached to a shape, as well as properties and methods for manipulating the text.

expression.**TextRange**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example adds text to the text frame of shape one in the active publication, and then formats the new text. This example assumes there is at least one shape on the first page of the active publication.

```
Sub AddTextToTextFrame()
    With ActiveDocument.Pages(1).TextFrame.TextRange
        .Text = "My Text"
        With .Font
            .Bold = msoTrue
            .Size = 25
            .Name = "Arial"
        End With
    End With
End Sub
```

The following example adds a rectangle to the active publication and adds text to it.

```
Sub AddTextToShape()
    With ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoShapeRecta
        Left:=72, Top:=72, Width:=250, Height:=140)
        .TextFrame.TextRange.Text = "Here is some test text"
    End With
End Sub
```

TextStyle Property

Returns or sets a **Variant** that represents the text style applied to a paragraph. Read/write.

expression.**TextStyle**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example changes the text style of the selection if the selection isn't formatted with the Normal text style. This example assumes text is selected in the active publication.

```
Sub SetTextStyle()
    With Selection.TextRange.ParagraphFormat
        If .TextStyle <> "Normal" Then _
            .TextStyle = "Normal"
    End With
End Sub
```

TextStyles Property

Returns a **TextStyles** collection that contains a publication's text styles.

expression.**TextStyles**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the style name and base style of the first style in the **TextStyles** collection.

```
Sub BaseStyleName()
    With ActiveDocument.TextStyles(1)
        MsgBox "Style name= " & .Name _
            & vbCr & "Base style= " & .BaseStyle
    End With
End Sub
```

TextToDisplay Property

Returns or sets a **String** that represents the text displayed for a hyperlink. Read/write.

expression.**TextToDisplay**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets the hyperlink display text and address of the first hyperlink on the first page. This example assumes the first page of the active publication contains at least one shape with at least one text hyperlink.

```
Sub SetHyperlinkTextToDisplay()
    With ActiveDocument.Pages(1).Shapes(1) _
            .TextFrame.TextRange.Hyperlinks.Item(1)
        .TextToDisplay = "Tailspin Toys Web Site"
        .Address = "http://www.tailspintoys.com/"
    End With
End Sub
```

TextureName Property

Returns a **String** indicating the name of the custom texture file for the specified fill. Read-only.

expression.**TextureName**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **UserTextured** method to set the texture file for the fill.

Example

This example adds an oval to the active publication. If shape one on the active publication has a fill with a user-defined texture, the new oval will have the same fill as shape one. If shape one has any other type of fill, the new oval will have a green marble fill.

```
Dim ffNew As FillFormat

With ActiveDocument.Pages(1).Shapes
    Set ffNew = .AddShape(Type:=msoShapeOval, _
        Left:=0, Top:=0, Width:=200, Height:=90).Fill

    With .Item(1).Fill
        If .Type = msoFillTextured And _
                .TextureType = msoTextureUserDefined Then
            ffNew.UserTextured _
                TextureFile:=.TextureName
        Else
            ffNew.PresetTextured _
                PresetTexture:=msoTextureGreenMarble
        End If
    End With
End With
```


Show All

TextureType Property

Returns an **MsoTextureType** constant indicating the texture type for the specified fill. Read-only.

MsoTextureType can be one of these MsoTextureType constants.

msoTexturePreset The fill uses a preset texture type.

msoTextureTypeMixed Indicates a combination of texture types for the specified shape range.

msoTextureUserDefined The fill uses a user-defined texture type.

expression.**TextureType**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is read-only. Use the **PresetTextured** or **UserTextured** method to set the texture type for the fill.

Example

This example applies a canvas texture to the fill for all shapes on the first page of the active publication that currently have fills with a user-defined texture.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop.Fill
        If .TextureType = msoTextureUserDefined Then
            .PresetTextured _
                PresetTexture:=msoTextureCanvas
        End If
    End With
Next shpLoop
```

TextWrap Property

Returns a **WrapFormat** object that represents the properties for wrapping text around a shape or shape range.

expression.**TextWrap**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example adds an oval to the active publication and specifies that publication text wrap around the left and right sides of the square that circumscribes the oval. There will be a 0.1-inch margin between the publication text and the top, bottom, left side, and right side of the square.

```
Sub SetTextWrapFormatProperties()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes.AddShape(Type:=msoS
         Left:=36, Top:=36, Width:=100, Height:=35)
    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
        .DistanceAuto = msoFalse
        .DistanceTop = InchesToPoints(0.1)
        .DistanceBottom = InchesToPoints(0.1)
        .DistanceLeft = InchesToPoints(0.1)
        .DistanceRight = InchesToPoints(0.1)
    End With
End Sub
```

ThreeD Property

Returns a **ThreeDFormat** object.

expression.**ThreeD**

expression Required. An expression that returns one of the objects in the Applies To list.

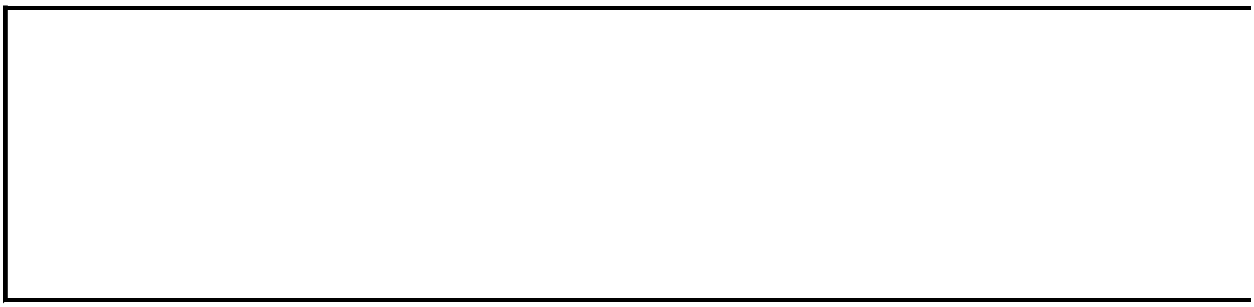
Remarks

Use the **ThreeD** property to return a **ThreeDFormat** object whose properties are used to format the 3-D appearance of the specified shape.

Example

This example sets the depth, extrusion color, extrusion direction, and lighting direction for the 3-D effects applied to shape one in the active publication.

```
Dim tdfTemp As ThreeDFormat

Set tdfTemp = _
    ActiveDocument.Pages(1).Shapes(1).ThreeD

With tdfTemp
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)
    .SetExtrusionDirection _
        PresetExtrusionDirection:=msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```



TintAndShade Property

Returns or sets a **Single** that represents the lightening or darkening of a specified shape's color. Read/write.

expression.**TintAndShade**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

You can enter a number from -1 (darkest) to 1 (lightest) for the **TintAndShade** property, 0 (zero) being neutral.

Example

This example creates a new shape in the active document, sets the fill color, and lightens the color shade.

```
Sub NewTintedShape()
    Dim shpHeart As Shape
    Set shpHeart = ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeHeart, Left:=150, _
        Top:=150, Width:=250, Height:=250)
    With shpHeart.Fill.ForeColor
        .CMYK.SetCMYK Cyan:=255, Magenta:=28, Yellow:=0, Black:=0
        .TintAndShade = 0.3
    End With
End Sub
```


Show All

Top Property

 **Top** property as it applies to the **ReaderSpread** object.

Returns the a **Single** that represents the distance (in points) from the top edge of the workspace to the top edge of the page. Read-only.

expression.**Top**

expression Required. An expression that returns one of the above objects.

 **Top** property as it applies to the **PrintableRect** object.

Returns the a **Single** that represents the distance (in points) from the top edge of the printer sheet to the top edge of the printable rectangle. Read-only.

expression.**Top**

expression Required. An expression that returns one of the above objects.

 **Top** property as it applies to the **Window** object.

Returns or sets a **Long** that represents the distance between the top edge of the screen and the application window. Read/write.

expression.**Top**

expression Required. An expression that returns one of the above objects.

 **Top** property as it applies to the **Shape** object.

Returns or sets a **Variant** that represents the distance between the top of the page and the top of a shape. Read/write.

expression.**Top**

expression Required. An expression that returns one of the above objects.

Top property as it applies to the **ShapeRange** object.

Returns a **Variant** that represents the distance between the top of the page and the top shape in a range of shapes. Read-only.

expression.**Top**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **Window** object.

This example verifies that the state of application window is neither maximized nor minimized and then resizes the window and moves it to 150 points from the top of the screen.

```
Sub MoveWindow()
    With ActiveWindow
        If .WindowState = pbWindowStateNormal Then
            .Top = 150
            .Resize Width:=500, Height:=500
        End If
    End With
End Sub
```

As it applies to the **Shape** object.

This example changes the position, size, and type of shape of the first shape on the first page of the active publication. This example assumes there is at least one shape on the first page of the active publication.

```
Sub MoveSizeChangeShape()
    With ActiveDocument.Pages(1).Shapes(1)
        .Top = 72
        .Left = 72
        .Width = 150
        .Height = 150
        .AutoShapeType = msoShape5pointStar
    End With
End Sub
```

TopMargin Property

Returns or sets a **Variant** that represents the distance (in points) between the top edge of the printer sheet and the top edge of the publication pages. Read/write.

expression.**TopMargin**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

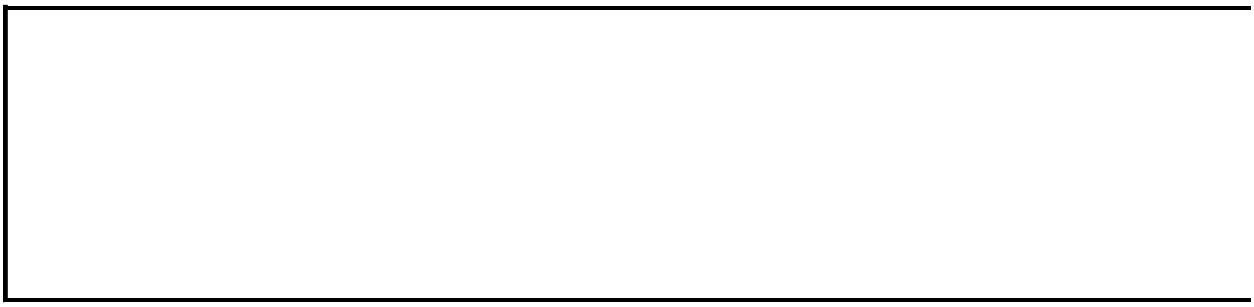
You can only use the **TopMargin** property when printing multiple pages on a single sheet of printer paper.

When used with the **Label** object, the **TopMargin** property is read/write only when accessed from **.PageSetup.Label**. Otherwise, it is read-only.

Example

This example specifies margins of a quarter of an inch between the top and left edges of the printer paper and the top and left edges of the pages in the active publication.

```
Sub SetTopMargin()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(5)
        .PageWidth = InchesToPoints(8)
        .MultiplePagesPerSheet = True
        .TopMargin = InchesToPoints(0.25)
        .LeftMargin = InchesToPoints(0.25)
    End With
End Sub
```



Tracking Property

Returns or sets a **Variant** indicating the tracking value used to display space between the characters in the specified text range. Read/write.

expression.**Tracking**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Valid range is 0.0 to 600.0 points. Setting the property to 0.0 disables tracking. Indeterminate values are returned as -2.

Example

This example disables tracking in the second story by setting the **Tracking** property to zero.

```
Sub DisableTracking()

    Application.ActiveDocument.Stories(2).TextRange.Font.Tracking =

End Sub
```


Show All

TrackingPreset Property

Returns or sets a **PbTrackingPresetType** constant representing the preset tracking type for characters in the specified font in a text range. Read/write.

PbTrackingPresetType can be one of these PbTrackingPresetType constants.

pbTrackingCustom
pbTrackingLoose
pbTrackingMixed
pbTrackingNormal
pbTrackingTight
pbTrackingVeryLoose
pbTrackingVeryTight

expression.**TrackingPreset**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Loose and very loose tracking leaves ample space between characters, whereas tight and very tight tracking can produce character overlap.

Example

This example specifies tight tracking as the preset for the characters in the second story.

```
Sub TrackingType()

    Application.ActiveDocument.Stories(2).TextRange _
        .Font.TrackingPreset = pbTrackingTight

End Sub
```

Transparency Property

Returns or sets a **Single** indicating the degree of transparency of the specified fill, shadow, or line as a value between 0.0 (opaque) and 1.0 (clear). Read/write.

expression.**Transparency**

expression Required. An expression that returns one of the objects in the Applies To list.

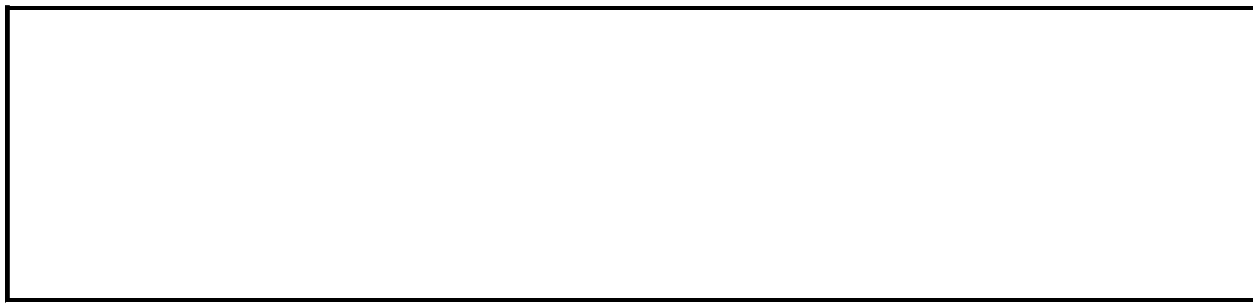
Remarks

The value of this property affects the appearance of solid-colored fills and lines only; it has no effect on the appearance of patterned lines or patterned, gradient, picture, or textured fills.

Example

This example sets the shadow for shape three in the active publication to semitransparent red. If the shape doesn't already have a shadow, this example adds one to it.

```
With ActiveDocument.Pages(1).Shapes(3).Shadow
    .Visible = True
    .ForeColor.RGB = RGB(255, 0, 0)
    .Transparency = 0.5
End With
```



TransparencyColor Property

Returns or sets an **MsoRGBType** constant that represents the transparency color. Read/write.

expression.**TransparencyColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a picture on the first page and sets the transparency color to black.

```
Sub SetTransparentColor()
    With ActiveDocument.Pages(1).Shapes.AddPicture( _
            FileName:="C:\My Pictures\Sample.gif", LinkToFile:=msoFa
            SaveWithDocument:=msoTrue, Left:=36, Top:=36)
        .PictureFormat.TransparencyColor = RGB(Red:=255, Green:=255,
    End With
End Sub
```


Show All

TransparentBackground Property

Returns or sets an **MsoTriState** constant indicating whether the parts of the specified picture that are defined as the transparent color appear transparent. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse Parts of the picture whose color is the transparency color do not appear transparent.

msoTriStateMixed Return value only. Indicates a combination of **msoTrue** and **msoFalse** for the specified objects.

msoTriStateToggle Set value only. Toggles between **msoTrue** and **msoFalse**.

msoTrue Parts of the picture whose color is the transparency color appear transparent.

expression.**TransparentBackground**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **TransparencyColor** property to set the transparent color.

This property applies to bitmaps only.

If you want to be able to see through the transparent parts of the picture all the way to the objects behind the picture, you must set the **Visible** property of the picture's **FillFormat** object to **mso False**. If your picture has a transparent color and the **Visible** property of the picture's **FillFormat** object is set to **msoTrue**, the picture's fill will be visible through the transparent color, but objects behind the picture will be obscured.

Example

This example sets the color blue as the transparent color for shape one in the active publication. For the example to work, shape one must be a bitmap.

```
With ActiveDocument.Pages(1).Shapes(1)

    With .PictureFormat
        .TransparentBackground = msoTrue
        ' RGB(0, 0, 255) is the color blue.
        .TransparencyColor = RGB(0, 0, 255)
    End With

    .Fill.Visible = False

End With
```


Show All

Type Property

 **Type** property as it applies to the **CalloutFormat** object.

Returns or sets an **MsoCalloutType** constant that represents the callout type. Read/write.

MsoCalloutType can be one of these MsoCalloutType constants.
msoCalloutFour
msoCalloutMixed
msoCalloutOne
msoCalloutThree
msoCalloutTwo

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **ColorFormat** object.

Returns or sets a **PbColorType** constant that represents the shape color type. Read-only.

PbColorType can be one of these PbColorType constants.
pbColorTypeCMYK
pbColorTypeInk
pbColorTypeRGB
pbColorTypeScheme

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **ConnectorFormat** object.

Returns or sets an **MsoConnectorType** constant that represents the connector type. Read/write.

MsoConnectorType can be one of these MsoConnectorType constants.
msoConnectorCurve
msoConnectorElbow
msoConnectorStraight
msoConnectorTypeMixed

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **Field** object.

Returns a **PbFieldType** constant that represents the field type. Read-only.

PbFieldType can be one of these PbFieldType constants.
pbFieldDateTime
pbFieldHyperlinkAbsolutePage
pbFieldHyperlinkEmail
pbFieldHyperlinkFile
pbFieldHyperlinkRelativePage
pbFieldHyperlinkURL
pbFieldIHIV
pbFieldMailMerge
pbFieldNone
pbFieldPageNumber
pbFieldPageNumberNext
pbFieldPageNumberPrev
pbFieldPhoneticGuide
pbFieldWizardSampleText

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **FillFormat** object.

Returns an **MsoFillType** constant that represents the fill format type of a shape. Read-only.

MsoFillType can be one of these MsoFillType constants.
msoFillBackground
msoFillGradient
msoFillMixed
msoFillPatterned
msoFillPicture
msoFillSolid
msoFillTextured

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **Hyperlink** object.

Returns an **MsoHyperlinkType** constant that represents the hyperlink type. Read-only.

MsoHyperlinkType can be one of these MsoHyperlinkType constants.
msoHyperlinkInlineShape
msoHyperlinkRange
msoHyperlinkShape

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **MailMergeDataSource** object.

Returns a **Long** that represents the type of mail merge or catalog merge data source. Read-only.

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **RulerGuide** object.

Returns or sets a **PbRulerGuideType** constant that represents the ruler guide type. Read/write.

PbRulerGuideType can be one of these PbRulerGuideType constants.
pbRulerGuideTypeHorizontal
pbRulerGuideTypeVertical

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **Selection** object.

Returns a **PbSelectionType** constant that represents the selection type. Read-only.

PbSelectionType can be one of these PbSelectionType constants.
pbSelectionNone
pbSelectionShape
pbSelectionShapeSubSelection
pbSelectionTableCells
pbSelectionText

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **ShadowFormat** object.

Returns or sets an **MsoShadowType** constant that represents the shadow type of a shape. Read/write.

MsoShadowType can be one of these MsoShadowType constants.
msoShadow1
msoShadow10
msoShadow11
msoShadow12
msoShadow13
msoShadow14
msoShadow15
msoShadow16
msoShadow17
msoShadow18
msoShadow19
msoShadow2
msoShadow20
msoShadow3
msoShadow4
msoShadow5
msoShadow6
msoShadow7
msoShadow8
msoShadow9
msoShadowMixed

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **Shape** object and the **ShapeRange** object.

Returns a **PbShapeType** constant that represents the shape type. Read-only.

PbShapeType can be one of these PbShapeType constants.
pbAutoShape
pbCallout

pbCatalogMergeArea
pbChart
pbComment
pbEmbeddedOLEObject
pbFormControl
pbFreeform
pbGroup
pbGroupWizard
pbLine
pbLinkedOLEObject
pbLinkedPicture
pbMedia
pbOLEControlObject
pbPicture
pbPlaceholder
pbShapeTypeMixed
pbTable
pbTextEffect
pbTextFrame
pbWebCheckBox
pbWebCommandButton
pbWebHTMLFragment
pbWebListBox
pbWebNavigationBar
pbWebMultiLineTextBox
pbWebOptionButton
pbWebSingleLineTextBox
pbWebWebComponent
pbWebWebNavigationBar

Note There can be only one shape of type **pbCatalogMergeArea** for a given publication page. If a shape is a catalog merge area, the following methods return "Permission Denied": **Apply**, **Copy**, **Cut**, **Duplicate**, **Flip**, **IncrementLeft**, **IncrementRotation**, **IncrementTop**, **PickUp**,

RerouteConnections, **SetShapesDefaultProperties**, and **Ungroup**.

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **Story** object.

Returns a **PbStoryType** constant that represents the type of story. Read-only.

PbStoryType can be one of these PbStoryType constants.
pbStoryContinuedFrom
pbStoryContinuedOn
pbStoryTable
pbStoryTextFrame

expression.**Type**

expression Required. An expression that returns one of the above objects.

 **Type** property as it applies to the **WrapFormat** object.

Returns a **PbWrapType** constant that represents how text wraps around the specified shape. Read/write.

PbWrapType can be one of these PbWrapType constants.
pbWrapTypeMixed
pbWrapTypeNone
pbWrapTypeSquare
pbWrapTypeThrough
pbWrapTypeTight

expression.**Type**

expression Required. An expression that returns one of the above objects.

Example

 As it applies to the **Callout** and **Shape** objects.

This example formats the callout type of the specified shape if the shape is a callout. This example assumes there is at least one shape on the first page of the active publication.

```
Sub SetCalloutType()
    With ActiveDocument.Pages(1).Shapes(1)
        If .Type = pbCallout Then
            With .Callout
                .Border = msoTrue
                .Type = msoCalloutThree
            End With
        End If
    End With
End Sub
```

 As it applies to the **WrapFormat** object.

The following example adds an oval to the active publication and specifies that the publication text wrap around both the left and right sides of the square that surrounds the oval.

```
Sub SetTextWrapType()
    Dim shpOval As Shape

    Set shpOval = ActiveDocument.Pages(1).Shapes.AddShape( _
        Type:=msoShapeOval, Left:=36, Top:=36, _
        Width:=100, Height:=35)

    With shpOval.TextWrap
        .Type = pbWrapTypeSquare
        .Side = pbWrapSideBoth
    End With
End Sub
```

 As it applies to the **Selection** object.

This example checks to see if the selection is text and if it is, makes the selected text bold.

```
Sub IfCellData()
    Dim rowTable As Row
    If Selection.Type = pbSelectionText Then
        Selection.TextRange.Font.Bold = msoTrue
    End If
End Sub
```

TypeNReplace Property

True for Publisher to replace unreadable Asian character clusters resulting from invalid keyboard sequences. Read/write **Boolean**.

expression.**TypeNReplace**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example instructs Publisher to replace unreadable Asian character clusters resulting from invalid keyboard sequences.

```
Sub TypeReplace()
    Options.TypeNReplace = True
End Sub
```


Show All

Underline Property

Returns or sets an **PbUnderlineType** constant that indicates the type of underline for the selected characters in the specified font in a text range. Read/write.

PbUnderlineType can be one of these PbUnderlineType constants.
pbUnderlineDash
pbUnderlineDashHeavy
pbUnderlineDashLong
pbUnderlineDashLongHeavy
pbUnderlineDotDash
pbUnderlineDotDashHeavy
pbUnderlineDotDotDash
pbUnderlineDotDotDashHeavy
pbUnderlineDotHeavy
pbUnderlineDotted
pbUnderlineDouble
pbUnderlineMixed
pbUnderlineNone
pbUnderlineSingle
pbUnderlineThick
pbUnderlineWavy
pbUnderlineWavyDouble
pbUnderlineWavyHeavy
pbUnderlineWordsOnly

expression.**Underline**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example formats the characters of the first story with a dashed and heavy underline.

```
Sub DashHeavy()

    Application.ActiveDocument.Stories(1).TextRange _
        .Font.Underline = pbUnderlineDashHeavy

End Sub
```

UndoActionsAvailable Property

Returns the number of actions available on the undo stack. Read-only **Long**.

expression.**UndoActionsAvailable**

expression Required. An expression that returns a **Document** object.

Example

The following example adds a rectangle that contains a text frame to the fourth page of the active publication. Some font properties and the text of the text frame are set. A test is then run to determine whether the font in the text frame is Courier. If so, the **Undo** method is used with the value of the **UndoActionsAvailable** property passed as a parameter to specify that all previous actions be undone.

The **Redo** method is then used with the value of the **RedoActionsAvailable** property minus 2 passed as a parameter to redo all actions except for the last two. A new font is specified for the text in the text frame, in addition to new text.

This example assumes the active document contains at least four pages.

```
Dim thePage As page
Dim theShape As Shape
Dim theDoc As Publisher.Document

Set theDoc = ActiveDocument
Set thePage = theDoc.Pages(4)

With theDoc
    With thePage
        Set theShape = .Shapes.AddShape(msoShapeRectangle, _
            75, 75, 190, 30)
        With theShape.TextFrame.TextRange
             .Font.Size = 12
             .Font.Name = "Courier"
             .Text = "This font is Courier."
        End With
     End With

    If thePage.Shapes(1).TextFrame.TextRange.Font.Name = "Courier" T
        .Undo (.UndoActionsAvailable)
        .Redo (.RedoActionsAvailable - 2)
        With theShape.TextFrame.TextRange
            .Font.Name = "Verdana"
            .Text = "This font is Verdana."
        End With
    End If
End With
```

UpdatePersonalInfoOnSave Property

Returns or sets a **Boolean** indicating whether to update personal information stored with a publication when it is saved. Read/write.

expression.**UpdatePersonalInfoOnSave**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Caution Use this property with caution. Sensitive or confidental information could be revealed to other users.

Use the **RemovePersonalInformation** property to remove personal information from a publication when it is saved.

Example

The following example sets Publisher to update personal information in all publications when they are saved.

```
Options.UpdatePersonalInfoOnSave = True
```

UseCatalogAtStartup Property

True for Microsoft Publisher to show the catalog when starting up. Read/write **Boolean**.

expression.**UseCatalogAtStartup**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets global options for Microsoft Publisher, including not displaying the catalog upon startup.

```
Sub SetGlobalOptions()
    With Options
        .AutoFormatWord = True
        .AutoKeyboardSwitching = True
        .AutoSelectWord = True
        .DragAndDropText = True
        .UseCatalogAtStartup = False
        .UseHelpfulMousePointers = False
    End With
End Sub
```


Show All

UseCharBasedFirstLineIndent Property

Returns or sets an **MsoTriState** constant that specifies whether a paragraph is indented using East Asian character width. Read/write.

MSOTriState can be one of these **MSOTriState** constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**UseCharBasedFirstLineIndent**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The value of **UseCharBasedFirstLineIndent** can be set only if East Asian languages are enabled on the client computer, whereas the value can be returned regardless of whether East Asian languages are enabled. Note that **UseCharBasedFirstLineIndent** must be set before the **CharBasedFirstLineIndent** property can be returned or set. A run-time "permission denied" error is returned if **UseCharBasedFirstLineIndent** is not set first.

If **UseCharBasedFirstLineIndent** is **True**, the paragraph is indented using East Asian character width, and if it is **False** it is not. The default value is **False**.

Example

The following example creates a text box on the fourth page of the active publication. After the **UseCharBasedFirstLineIndent** property is set to **True**, the width of the first line indent is set to 15 points by using the **CharBasedFirstLineIndent** property. Font properties are then set, and text is inserted into the paragraph.

```
Dim theTextBox As Shape

Set theTextBox = ActiveDocument.Pages(4).Shapes _
        .AddShape(msoShapeRectangle, 100, 100, 300, 200)

With theTextBox
    .TextFrame.TextRange.ParagraphFormat _
        .UseCharBasedFirstLineIndent = msoTrue
    .TextFrame.TextRange.ParagraphFormat _
        .CharBasedFirstLineIndent = 15
    .TextFrame.TextRange.Font.Name = "Verdana"
    .TextFrame.TextRange.Font.Size = 12
    .TextFrame.TextRange.Text = "This is a test sentence." _
       & Chr(13) & "This is another test sentence."
End With
```

UseCustomHalftone Property

Returns or sets a **Boolean** that represents whether to use custom halftone settings. **True** to be able to specify custom halftone settings for any printable plate. **False** to use Publisher's default settings for all printable plates. Read/write.

expression.**UseCustomHalftone**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **UseCustomHalftone** property to be able to set the **Angle** and **Frequency** properties of any **PrintablePlate** object in a publication's **PrintablePlates** collection.

The property corresponds to the **Use Publisher defaults** and **Use custom settings** options on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

This example sets the spot color plates (plates five and higher) of a process and spot color publication to the same custom angle and frequency. The example assumes that the publication's color mode has been specified as process and spot colors, and the publication's print mode has been specified as separations.

```
Sub SetSpotColorPlatesProperties()

ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

Dim intCount As Integer

With ActiveDocument.AdvancedPrintOptions.PrintablePlates
    For intCount = 5 To .Count
        With .Item(intCount)
            .Angle = 45
            .Frequency = 150
        End With
    Next
End With

End Sub
```


Show All

UseDiacriticColor Property

Returns or sets **MsoTriState** constant indicating whether you can set the color of diacritics in the specified text range. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The color of diacritics cannot be set in the specified text range.

msoTriStateMixed Return value indicating a combination of **msoTrue** and **msoFalse** for the specified text range.

msoTriStateToggle Set value which toggles between **msoTrue** and **msoFalse**.

msoTrue The color of diacritics can be set in the specified text range.

expression.**UseDiacriticColor**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example test the text in the first story of the publication for the state of the **UseDiacriticColor** property. If it is **msoTrue**, then the **DiacriticColor** is set to blue otherwise, a message box is displayed.

```
Sub UseDiaColor()

    Dim fntDC As Font

    Set fntDC = Application.ActiveDocument.Stories(1).TextRange.Font
    If fntDC.UseDiacriticColor = msoTrue Then
        fntDC.DiacriticColor.RGB = RGB(Red:=0, Green:=0, Blue:=255)
    Else
        MsgBox "The UseDiacriticColor property is set to False"
    End If

End Sub
```

UseEnvelopePaperSizes Property

True to print envelopes using the envelope paper size. Read/write **Boolean**.

expression.**UseEnvelopePaperSizes**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Returns "Permission Denied" for publications that are not envelopes.

Example

This example sets Publisher's envelope printing options. This example assumes the publication is an envelope.

```
Sub SetEnvelopeOptions()
    With Options
        .UseEnvelopePrintOptions = True
        .UseEnvelopePaperSizes = True
    End With
End Sub
```

UseEnvelopePrintOptions Property

True to print envelopes using the envelope printing options. Read/write **Boolean**.

expression.**UseEnvelopePrintOptions**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Returns "Permission Denied" for publications that are not envelopes.

Example

This example sets Publisher's envelope printing options. This example assumes the publication is an envelope.

```
Sub SetEnvelopeOptions()
    With Options
        .UseEnvelopePrintOptions = True
        .UseEnvelopePaperSizes = True
    End With
End Sub
```

UseHelpfulMousePointers Property

True for Microsoft Publisher to display helpful mouse pointers. Read/write **Boolean**.

expression.**UseHelpfulMousePointers**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example sets global options for Microsoft Publisher, including disabling the display of helpful mouse pointers.

```
Sub SetGlobalOptions()
    With Options
        .AutoFormatWord = True
        .AutoKeyboardSwitching = True
        .AutoSelectWord = True
        .DragAndDropText = True
        .UseCatalogAtStartup = False
        .UseHelpfulMousePointers = False
    End With
End Sub
```

UseOnlyPublicationFonts Property

Returns or sets a **Boolean** that represents whether to only use publication fonts for printing the specified publication. **True** to print the specified publication using only fonts downloaded from your computer. Read/write. The default is **True**.

expression.**UseOnlyPublicationFonts**()

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

Publication fonts are fonts that are downloaded from your computer, as opposed to fonts residing at the printer or imagesetter.

Set this property to **False** to enable the printer to print the specified publication using its resident fonts (stored in ROM, RAM, or on a hard disk drive) that have the same name as the fonts downloaded from your computer.

Note This may result in the printer substituting resident printer for fonts downloaded from your computer. This results in a slightly faster print time. However, if the resident fonts are not exactly identical to your computer fonts (even if they have the same name), this may cause your printed publication to look different than expected.

Setting this property to **True** ensures that the fonts used to print the publication are the same ones used to create it.

This property corresponds to the **Fonts** controls on the **Graphics and Fonts** tab of the **Advanced Print Settings** dialog box.

Example

The following example tests to determine if the active publication will be printed using only publication fonts. If it will not, it is set to use only publication fonts.

```
Sub PrintWithPublicationFontsOnly()
   With ActiveDocument.AdvancedPrintOptions
      .UseOnlyPublicationFonts = True
   End With
End Sub
```


Show All

Value Property

 Value property as it applies to the **WebCheckBox** and **WebOptionButton** objects.

Returns or sets a **String** that represents the value of a Web check box or option button. Read/write.

expression.**Value**

expression Required. An expression that returns one of the above objects.

 Value property as it applies to the **MailMergeDataField** and **MailMergeMappedDataField** objects.

Returns a **String** that represents the value of a mail merge data field record or a mapped data field. Read-only.

expression.**Value**

expression Required. An expression that returns one of the above objects.

 Value property as it applies to the **Tag** object.

Returns or sets a **Variant** that represents the value of a tag of a shape, page, or publication. Read/write.

expression.**Value**

expression Required. An expression that returns one of the above objects.

Example

 As it applies to the **WebCheckBox** object.

This example creates a new Web check box control, assigns a name and value to it, and indicates its initial state is checked.

```
Sub CreateWebButton()
    With ActiveDocument.Pages(1).Shapes.AddWebControl _
            (Type:=pbWebControlCheckBox, Left:=72, _
            Top:=72, Width:=100, Height:=50)
        .Name = "ControlBox"
        With .WebCheckBox
            .Value = "This is a check box."
            .Selected = msoTrue
        End With
    End With
End Sub
```

 As it applies to the **Tag** object.

This example creates a new tag for the active publication and then displays the value of the tag.

```
Sub CreatePublicationTag()
    With ActiveDocument
        .Tags.Add Name:="ActivePub", Value:="This is the active publ
        MsgBox .Tags(1).Value
    End With
End Sub
```

Values Property

Returns a **WizardValues** collection representing all the valid values for a wizard property.

expression.**Values**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the current value for the first wizard property in the active publication and then lists all the other possible values.

```
Dim valAll As WizardValues
Dim valLoop As WizardValue

With ActiveDocument.Wizard
    Set valAll = .Properties(1).Values

    MsgBox "Wizard: " & .Name & vbLf & _
        "Property: " & .Properties(1).Name & vbLf & _
        "Current value: " & .Properties(1).CurrentValueId

    For Each valLoop In valAll
        MsgBox "Possible value: " & valLoop.ID & " (" & valLoop.Name
    Next valLoop
End With
```

Version Property

Returns a **String** indicating the version number of the currently-installed copy of Microsoft Publisher. Read-only.

expression.**Version**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example displays the version and build number of the currently-installed copy of Microsoft Publisher.

```
MsgBox "You are currently running Microsoft Publisher, " _
    & " version " & Application.Version & ", build " _
    & Application.Build & "."
```

VerticalBaseLineOffset Property

Returns a **Single** that represents the vertical baseline offset of the specified **LayoutGuides** object. Read/write.

expression.**VerticalBaseLineOffset**

expression Required. An expression that returns a **LayoutGuides** object.

Remarks

When setting the layout guide properties of a **Page** object it must be returned from the **MasterPages** collection.

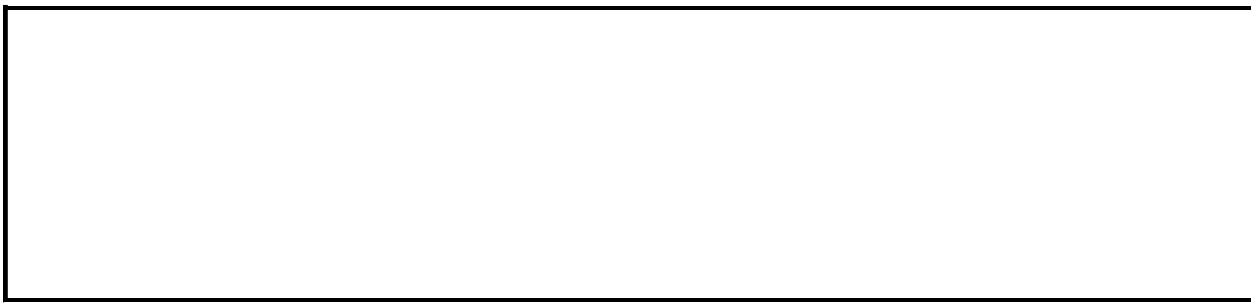
Example

This example sets the vertical baseline offset of the layout guides object to 12 for the second master page in the active document.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.MasterPages(2).LayoutGuides
objLayout.VerticalBaseLineOffset = 12
```

Setting the layout guide properties for the active document will only affect the first master page. This example sets the vertical baseline offset of the active document's layout guides to 12, affecting only the first master page.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.Pages(1).LayoutGuides
objLayout.VerticalBaseLineOffset = 12
```



VerticalBaseLineSpacing Property

Returns a **Single** that represents the vertical baseline spacing of the specified **LayoutGuides** object. Read/write.

expression.**VerticalBaseLineSpacing**

expression Required. An expression that returns a **LayoutGuides** object.

Remarks

When setting the layout guide properties of a **Page** object it must be returned from the **MasterPages** collection.

Example

This example sets the vertical baseline spacing of the **LayoutGuides** object to 12 for the second master page in the active document.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.MasterPages(2).LayoutGuides
objLayout.VerticalBaseLineSpacing = 12
```

This example sets the vertical baseline spacing of the active document's layout guides to 20, affecting only the first master page.

```
Dim objLayout As LayoutGuides
Set objLayout = ActiveDocument.LayoutGuides
objLayout.VerticalBaseLineSpacing = 20
```


Show All

VerticalFlip Property

 **VerticalFlip** as it applies to the **Shape** object and the **ShapeRange** object.

Returns **msoTrue** if the specified shape has been flipped around its vertical axis. Read-only **MsoTriState**.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The shape has not been flipped around its vertical axis.

msoTriStateMixed Indicates a combination of **msoTrue** and **msoFalse** for the specified shape range.

msoTriStateToggle Not used with this property.

msoTrue The shape has been flipped around its vertical axis.

expression.**VerticalFlip**

expression Required. An expression that returns one of the objects in the Applies To list.

 **VerticalFlip** property as it applies to the **AdvancedPrintOptions** object.

True to print a vertically mirrored image of the specified publication. The default is **False**. Read/write **boolean**.

expression.**VerticalFlip**

expression Required. An expression that returns an **AdvancedPrintOptions** object.

Remarks

This property is only accessible if the active printer is a PostScript printer. Returns a run-time error if a non-PostScript printer is specified. Use the **IsPostscriptPrinter** property of the **AdvancedPrintOptions** object to determine if the specified printer is a PostScript printer.

This property is saved as an application setting and applied to future instances of Publisher.

This property corresponds to the **Flip vertically** control on the **Page Settings** tab of the **Advanced Print Settings** dialog box.

This property is mostly used when printing to film on an imagesetter so that the image reads correctly when the emulsion side of the film is down (as when burning a press plate).

Example

 As it applies to the **Shape** and **ShapeRange** objects.

This example restores each shape on the active publication to its original state if it has been flipped horizontally or vertically.

```
Sub Flipper()

    Dim shpBall As Shape

    For Each shpBall In ActiveDocument.MasterPages.Item(1).Shapes
        If shpBall.HorizontalFlip = msoTrue Then shpBall.Flip msoFli
        If shpBall.VerticalFlip = msoTrue Then shpBall.Flip msoFlipV
    Next

End Sub
```

 As it applies to the **AdvancedPrintOptions** object.

The following example determines if the active printer is a PostScript printer. If it is, the active publication is set to print as a horizontally and vertically mirrored, negative image of itself.

```
Sub PrepToPrintToFilmOnImagesetter()

With ActiveDocument.AdvancedPrintOptions
    If .IsPostscriptPrinter = True Then
        .HorizontalFlip = True
        .VerticalFlip = True
        .NegativeImage = True
    End If
End With

End Sub
```

VerticalGap Property

When multiple pages are printed on one sheet of printer paper, returns or sets a **Variant** that represents the distance (in points) between the bottom edge of the publication page and top edge of the publication page in the row immediately below. Read/write.

expression.**VerticalGap**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

Use the **VerticalGap** property when printing multiple pages on a single sheet of printer paper. If the page size, including the values for the **VerticalGap** and **HorizontalGap** properties, is greater than half the paper size, Publisher will display an error.

When used with the **Label** object, the **VerticalGap** property is read/write only when accessed from **.PageSetup.Label**. Otherwise, it is read-only.

Example

This example sets the page height and width of the active document, specifies that it be printed with multiple pages on each sheet of printer paper, and sets the vertical gap between those two pages at half an inch. This example assumes the page orientation is set to landscape.

```
Sub SetVerticalGap()
    With ActiveDocument.PageSetup
        .PageHeight = InchesToPoints(8)
        .PageWidth = InchesToPoints(4)
        .MultiplePagesPerSheet = True
        .VerticalGap = InchesToPoints(0.5)
    End With
End Sub
```


Show All

VerticalPictureLocking Property

Returns or sets a **PbVerticalPictureLocking** constant indicating where newly inserted pictures appear in relation to the specified frame. Read/write.

PbVerticalPictureLocking can be one of these PbVerticalPictureLocking constants.

pbVerticalLockingBottom New pictures are inserted along the bottom edge of the frame.

pbVerticalLockingNone New pictures are inserted in the center between the top and bottom edges of the frame.

pbVerticalLockingStretch New pictures are vertically stretched to the full height of the frame.

pbVerticalLockingTop New pictures are inserted along the top edge of the frame.

expression.**VerticalPictureLocking**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example locks the specified picture to the top left corner of the picture frame. Shape one on page one of the active publication must be a picture frame for this example to work.

```
With ActiveDocument.Pages(1).Shapes(1).PictureFormat
    .HorizontalPictureLocking = pbHorizontalLockingLeft
    .VerticalPictureLocking = pbVerticalLockingTop
End With
```


Show All

VerticalRepeat Property

Returns a **Long** that represents the number of times the catalog merge area will repeat down the target publication page when the catalog merge is executed. Read-only.

expression.**VerticalRepeat**

expression Required. An expression that returns a **CatalogMergeShapes** object.

Remarks

When the catalog merge is executed, the catalog merge area repeats once for each selected record in the specified data source.

The number of times the catalog merge area repeats down the page is determined by the height of the area. Use the **Height** property of the **Shape** object to return or set the vertical size of the catalog merge area.

The **HorizontalRepeat** property of the **CatalogMergeShapes** object represents the number of times the catalog merge area repeats horizontally across the target publication page.

Example

The following example returns the number of times the catalog merge area will repeat horizontally and vertically on the target publication page when the catalog merge is performed. This example assumes the catalog merge area is the first shape on the first page of the specified publication.

```
Sub CatalogMergeDimensions()

    With ThisDocument.Pages(1).Shapes(1)
        Debug.Print .Width
        Debug.Print .CatalogMergeItems.HorizontalRepeat
        Debug.Print .Height
        Debug.Print .CatalogMergeItems.VerticalRepeat
    End With

End Sub
```

VerticalScale Property

Returns a **Long** that represents the scaling of the picture along its vertical axis. The scaling is expressed as a percentage (for example, 200 equals 200% scaling). Read-only.

expression.**VerticalScale**()

expression Required. An expression that returns a **PictureFormat** object.

Remarks

The effective resolution of a picture is inversely proportional to the scaling at which the picture is printed. The larger the scaling, the lower the effective resolution. For example, suppose a picture measuring 4 inches by 4 inches was originally scanned at 300 dpi. If that picture is scaled to 2 inches by 2 inches, its effective resolution is 600 dpi.

Use the **EffectiveResolution** property of the **PictureFormat** object to determine the resolution at which the picture or OLE object will print in the specified document.

Example

The following example prints selected image properties for each picture in the active publication.

```
Dim pgLoop As Page
Dim shpLoop As Shape

For Each pgLoop In ActiveDocument.Pages
    For Each shpLoop In pgLoop.Shapes
        If shpLoop.Type = pbPicture Or shpLoop.Type = pbLinkedPictur

            With shpLoop.PictureFormat

                  If .IsEmpty = msoFalse Then
                       Debug.Print "File Name: " & .Filename
                       Debug.Print "Resolution in Publication: " &
                       Debug.Print "Horizontal Scaling: " & .Horizo
                       Debug.Print "Height in publication: " & .Hei
                       Debug.Print "Vertical Scaling: " & .Vertical
                       Debug.Print "Width in publication: " & .Widt
                  End If

            End With
        End If
    Next shpLoop
Next pgLoop
```


Show All

VerticalTextAlignment Property

Returns or sets a **PbVerticalTextAlignmentType** constant that represents the vertical alignment of text in a text box. Read/write.

PbVerticalTextAlignmentType can be one of these PbVerticalTextAlignmentType constants.

pbVerticalTextAlignmentBottom

pbVerticalTextAlignmentCenter

pbVerticalTextAlignmentTop

expression.**VerticalTextAlignment**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example vertically centers the text in the specified text frame. This example assumes there is at least one shape on the first page of the active publication.

```
Sub SetVerticalAlignment()
    ActiveDocument.Pages(1).Shapes(1).TextFrame _
        .VerticalTextAlignment = pbVerticalTextAlignmentCenter
End Sub
```

Vertices Property

Returns the coordinates of the specified freeform drawing's vertices (and control points for Bézier curves) as a series of coordinate pairs. Read-only **Variant**.

expression.**Vertices**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

You can use the array returned by this property as an argument to the **AddCurve** or **AddPolyline** methods.

The following table shows how the **Vertices** property associates the values in the array `vertArray()` with the coordinates of a triangle's vertices.

vertArray element	Contains
`vertArray(1, 1)`	The horizontal distance from the first vertex to the left side of the page.
`vertArray(1, 2)`	The vertical distance from the first vertex to the top of the page.
`vertArray(2, 1)`	The horizontal distance from the second vertex to the left side of the page.
`vertArray(2, 2)`	The vertical distance from the second vertex to the top of the page.
`vertArray(3, 1)`	The horizontal distance from the third vertex to the left side of the page.
`vertArray(3, 2)`	The vertical distance from the third vertex to the top of the page.

Example

This example assigns the vertex coordinates for shape one in the active publication to the array variable `vertArray()` and displays the coordinates for the first vertex.

```
Dim vertArray As Variant
Dim sngX1 As Single
Dim sngY1 As Single

With ActiveDocument.Pages(1).Shapes(1)
    vertArray = .Vertices
    sngX1 = vertArray(1, 1)
    sngY1 = vertArray(1, 2)
    MsgBox "First vertex coordinates: " & sngX1 & ", " & sngY1
End With
```

This example creates a curve that has the same geometric description as shape one in the active publication. Shape one must contain 3*n*+1 vertices for this example to work, where *n* is an integer greater than or equal to 1.

```
With ActiveDocument.Pages(1).Shapes
    .AddCurve SafeArrayOfPoints:=.Item(1).Vertices
End With
```

ViewBoundariesAndGuides Property

Returns **True** if boundaries and guides are visible in the specified publication. Read/write **Boolean**.

expression.**ViewBoundariesAndGuides**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example opens a message box and displays if the current publication shows boundaries and guides.

```
Sub ViewBandG()

    MsgBox "Boundaries & Guides = " & _
    Application.ActiveDocument.ViewBoundariesAndGuides

End Sub
```

ViewHorizontalBaseLineGuides Property

Sets or returns a **Boolean** that represents whether or not the horizontal baseline guides are visible in the specified **Document** object. **True** if they are visible. **False** if they are not visible. Read/write.

expression.

expression Required. An expression that returns a **Document** object.

Remarks

The default setting for this property is **False**.

Example

The following example makes the horizontal baseline guides visible in the active document.

```
Dim objDocument As Document
Set objDocument = ActiveDocument
objDocument.ViewHorizontalBaseLineGuides = True
```

ViewMailMergeFieldCodes Property

True if merge field names are displayed in a mail merge publication; **False** if information from the current data record is displayed. Read/write **Boolean**.

expression.**ViewMailMergeFieldCodes**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

If the active publication isn't a mail merge publication, using this property has no effect.

Example

This example hides the mail merge field codes in the active publication.

```
ActiveDocument.MailMerge.ViewMailMergeFieldCodes = False
```

ViewTwoPageSpread Property

Returns **True** if the specified publication should be viewed as a two-page spread. Read/write **Boolean**.

expression.**ViewTwoPageSpread**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example opens a message box and displays if the current publication should be viewed in the in the two page spread mode.

```
Sub ViewTwoPage()

    MsgBox "View Two Page Spread = " & _
        Application.ActiveDocument.ViewTwoPageSpread

End Sub
```

ViewVerticalBaseLineGuides Property

Sets or returns a **Boolean** that represents whether or not the vertical baseline guides are visible in the specified **Document** object. **True** if they are visible. **False** if they are not visible. Read/write.

expression.**ViewVerticalBaseLineGuides**

expression Required. An expression that returns a **Document** object.

Remarks

The default setting for this property is **False**.

Example

The following example makes the vertical baseline guides visible in the active document.

```
Dim objDocument As Document
Set objDocument = ActiveDocument
objDocument.ViewVerticalBaseLineGuides = True
```


Show All

Visible Property

 Visible property as it applies to the **FillFormat**, **LineFormat**, **ShadowFormat**, and **ThreeDFormat** objects.

Returns or sets an **MsoTriState** constant indicating whether the specified object or the formatting applied to the specified object is visible. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse The specified object or formatting is not visible.

msoTriStateMixed Return value only. The specified shape range contains both objects with visible formatting and objects with invisible formatting.

msoTriStateToggle Set value only. Toggles the specified object between visible and invisble.

msoTrue The specified object or formatting is visible.

expression.**Visible**

expression Required. An expression that returns one of the above objects.

 Visible property as it applies to the **Window** object.

True if the window is visible. Read/write **Boolean**.

expression.**Visible**

expression Required. An expression that returns one of the above objects.

Example

As it applies to the **FillFormat**, **LineFormat**, **ShadowFormat**, and **ThreeDFormat** objects.

This example sets the horizontal and vertical offsets for the shadow of shape three on the first page in the active publication. The shadow is offset 5 points to the right of the shape and 3 points above it. If the shape doesn't already have a shadow, this example adds one to it.

```
With ActiveDocument.Pages(1).Shapes(3).Shadow
    .Visible = msoTrue
    .OffsetX = 5
    .OffsetY = -3
End With
```

As it applies to the **Window** object.

This example hides the Publisher window.

```
ActiveWindow.Visible = False
```

WebCheckBox Property

Returns the **WebCheckBox** object associated with the specified shape.

expression.**WebCheckBox**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web check box and specifies that its default state is checked.

```
Dim shpNew As Shape
Dim wcbTemp As WebCheckBox

Set shpNew = ActiveDocument.Pages(1).Shapes _
    .AddWebControl(Type:=pbWebControlCheckBox, Left:=100, _
    Top:=123, Width:=17, Height:=12)

Set wcbTemp = shpNew.WebCheckBox

wcbTemp.Selected = msoTrue
```

WebCommandButton Property

Returns the **WebCommandButton** object associated with the specified shape.

expression.**WebCommandButton**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a Web form Submit command button and sets the script path and file name to run when a user clicks the button.

```
Dim shpNew As Shape
Dim wcbTemp As WebCommandButton

Set shpNew = ActiveDocument.Pages(1).Shapes.AddWebControl _
    (Type:=pbWebControlCommandButton, Left:=150, _
    Top:=150, Width:=75, Height:=36)

Set wcbTemp = shpNew.WebCommandButton

With wcbTemp
    .ButtonText = "Submit"
    .ButtonType = pbCommandButtonSubmit
    .ActionURL = "http://www.tailspintoys.com/" _
        & "scripts/ispscript.cgi"
End With
```

WebComponentFormat Property

Returns the **WebComponentFormat** object associated with the specified shape.

expression.**WebComponentFormat**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The **WebComponentFormat** object's functionality is not accessible through Microsoft Visual Basic. It is not recommended to access the object using this property.

Example

The following example assigns an object variable to the **WebComponentFormat** object associated with the first shape on page one of the active publication.

```
Dim wcfTemp As Object

Set wcfTemp = ActiveDocument.Pages(1) _
    .Shapes(1).WebComponentFormat
```

WebListBox Property

Returns the **WebListBox** object associated with the specified shape.

expression.**WebListBox**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web list box and adds several items to it. Note that when initially created, a Web list box control contains three default items. This example includes a loop that deletes the default list box items before adding new items.

```
Dim shpNew As Shape
Dim wlbTemp As WebListBox
Dim intCount As Integer

Set shpNew = ActiveDocument.Pages(1).Shapes _
    .AddWebControl(Type:=pbWebControlListBox, Left:=100, _
    Top:=150, Width:=300, Height:=72)

Set wlbTemp = shpNew.Web ListBox

With wlbTemp
    .MultiSelect = msoFalse

    With .ListBoxItems
        For intCount = 1 To .Count
            .Delete (1)
        Next intCount

        .AddItem Item:="Green"
        .AddItem Item:="Purple"
        .AddItem Item:="Red"
        .AddItem Item:="Black"
    End With
End With
```

WebNavigationBarSetName Property

Returns a **String** that represents the name of the Web navigation bar set the specified shape is an instance of. Read-only.

expression.**WebNavigationBarSetName**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

This property is only accessible for shapes that represent an instance of a Web navigation bar set. Use the **Type** property of the **Shape** object to determine if a shape represents an instance of a Web navigation bar set.

Use the **WebNavigationBarSetName** property to return the name of a **WebNavigationBarSet** object. Multiple pages in a Web publication can each have a shape representing an instance of the same Web navigation bar set. Changes made to a **WebNavigationBarSet** object are reflected in all the shapes representing instances of that Web navigation bar set.

Example

The following example tests to determine which shapes on the first page of the active document represent instances of Web navigation bars. For each such shape found, the Web navigation bar it represents an instance of is set to auto update.

```
Sub SetWebBarsToAutoUpdate()

Dim shpLoop As Shape
Dim strWebNavBarName As String

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    If shpLoop.Type = pbWebNavigationBar Then

        strWebNavBarName = shpLoop.WebNavigationBarSetName
            With ActiveDocument.WebNavigationBarSets(strWebNavBarNam
                .AutoUpdate = True
            End With

    End If
Next

End Sub
```

WebNavigationBarSets Property

Returns a **WebNavigationBarSets** object representing a collection of all **WebNavigationBarSet** objects in the specified document. Read-only.

expression.**WebNavigationBarSets**

expression Required. An expression that returns a **Document** object.

Example

The following example sets an object variable to the collection of Web navigation bar sets in the active document and adds a new navigation bar set to it.

```
Dim objWebNavBarSets As WebNavigationBarSets

Set objWebNavBarSets = ActiveDocument.WebNavigationBarSets
objWebNavBarSets.AddSet _
    Name:="WebNavBarSet1", _
    Design:=pbnbDesignBracket, _
    AutoUpdate:=True
```

WebOptionButton Property

Returns the **WebOptionButton** object associated with the specified shape.

expression.**WebOptionButton**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web option button and specifies that its default state is selected.

```
Dim shpNew As Shape
Dim wobTemp As WebOptionButton

Set shpNew = ActiveDocument.Pages(1).Shapes.AddWebControl _
    (Type:=pbWebControlOptionButton, Left:=100, _
    Top:=123, Width:=16, Height:=10)

Set wobTemp = shpNew.WebOptionButton

wobTemp.Selected = msoTrue
```

WebOptions Property

Returns a **WebOptions** object, which represents the properties of Web publications. Read-only.

expression.**WebOptions**

expression Required. An expression that returns an **Application** object.

Example

The following example specifies that Web publications should not always be saved in default encoding, and that the encoding should be Unicode (UTF-8).

```
With Application.WebOptions
    .AlwaysSaveInDefaultEncoding = False
    .Encoding = msoEncodingUTF8
End With
```

WebPageOptions Property

Returns a **WebPageOptions** object, which represents the properties of a single Web page within a Web publication. Read-only.

expression.**WebPageOptions**

expression Required. An expression that returns a **Page** object.

Example

The following example sets the description and the background sound for the fourth page of the active Web publication.

```
With ActiveDocument.Pages(4).WebPageOptions
    .Description = "Company Profile"
    .BackgroundSound = "C:\CompanySounds\corporate_jingle.wav"
End With
```

WebTextBox Property

Returns the **WebTextBox** object associated with the specified shape.

expression.**WebTextBox**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a new Web text box, specifies default text, indicates that entry is required, and limits entry to 50 characters.

```
Dim shpNew As Shape
Dim wtbTemp As WebTextBox

Set shpNew = ActiveDocument.Pages(1).Shapes _
    .AddWebControl(Type:=pbWebControlSingleLineTextBox, _
    Left:=100, Top:=100, Width:=150, Height:=15)

Set wtbTemp = shpNew.WebTextBox

With wtbTemp



.DefaultText = "Please Enter Your Full Name"
    .RequiredControl = msoTrue
    .Limit = 50
End With
```

Weight Property

Returns or sets a **Variant** indicating the thickness of the specified line or cell border. Read/write.

expression.**Weight**

expression Required. An expression that returns one of the objects in the Applies To list.

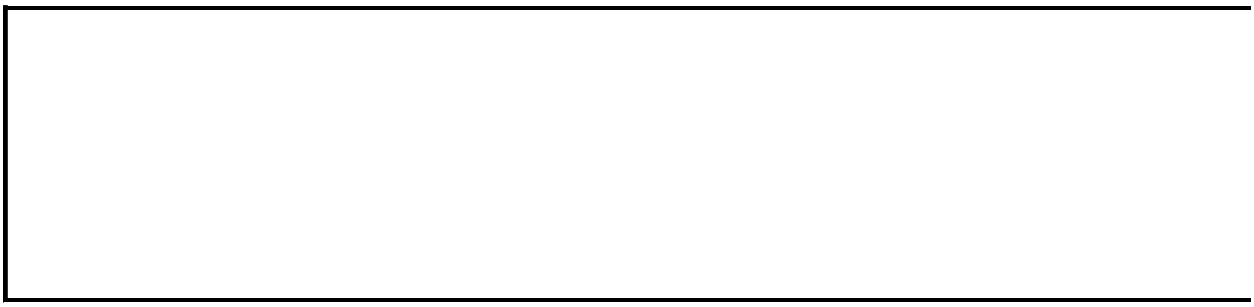
Remarks

Return values are in points. When setting the property, numeric values are evaluated in points, and strings can be in any units supported by Publisher (for example, "2.5 in").

Example

This example adds a green dashed line, two points thick, to the active publication.

```
With ActiveDocument.Pages(1).Shapes _
        .AddLine(BeginX:=10, BeginY:=10, _
        EndX:=250, EndY:=250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(0, 255, 255)
    .Weight = 2
End With
```




Show All

WidowControl Property

Sets or returns an **msoTriState** that represents whether or not the first or last line of the specified paragraph can appear by itself in a text box. Read/write.

msoCTrue
msoFalse The first or last line may appear by itself in a text box.
msoTriStateMixed
msoTriStateToggle
msoTrue The first or last line will not appear by itself in a text box.

expression.**WidowControl**

expression Required. An expression that returns a **ParagraphFormat** object.

Remarks

This option ensures that the first or last line of the specified paragraph will not appear by itself in a text frame. For example, if the last line in a specified paragraph is the first line of a widow controlled paragraph, a second line will be moved to the next text frame with it.

The default setting for this property is **msoFalse**.

Example

This example sets the **WidowControl** property to **msoTrue** for the specified **ParagraphFormat** object.

```
Dim objParaForm As ParagraphFormat
Set objParaForm = ActiveDocument.Pages(1).Shapes(1) _
    .TextFrame.TextRange.Paragraphs(1).ParagraphFormat
objParaForm.WidowControl = msoTrue
```


Show All

Width Property

 **Width** property as it applies to the **ReaderSpread** object and the **PrintableRect** object.

Returns a **Single** that represents the width, in points, of the page (for the **ReaderSpread** object) or the printable rectangle (for the **PrintableRect** object). Read-only.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **Label** object.

Returns or sets a **Variant** that represents the width (in points) of the label. Read-only.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **Window** object.

Returns or sets a **Long** that represents the width (in points) of the window. Read/write.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **Cell**, **CellRange**, and **Page** objects.

Returns a **Long** that represent the width (in points) of a cell, range of cells, or page. Read-only.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **Column** and **Shape** objects.

Returns or sets a **Variant** that represents the width (in points) of a specified table column or shape. Read/write.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **ShapeRange** object.

Returns a **Variant** that represents the width (in points) of a specified range of shapes. Read-only.

expression.**Width**

expression Required. An expression that returns one of the above objects.

 **Width** property as it applies to the **PictureFormat** object.

Returns a **Variant** that represents the width, in points, of the specified picture. Read-only.

expression.**Width**

expression Required. An expression that returns a **PictureFormat** object.

Example

As it applies to the **Window** object.

This example sets the height and width of the active window if the window is neither maximized nor minimized.

```
Sub SetWindowHeight()
    With ActiveWindow
        If .WindowState = pbWindowStateNormal Then
            .Height = InchesToPoints(5)
            .Width = InchesToPoints(5)
        End If
    End With
End Sub
```

As it applies to the **Column** object.

This example creates a new table and sets the height and width of the second row and column, respectively.

```
Sub SetRowHeightColumnWidth()
    With ActiveDocument.Pages(1).Shapes.AddTable(NumRows:=3, _
            NumColumns:=3, Left:=80, Top:=80, Width:=400, Height:=12
            .Rows(2).Height = 72
            .Columns(2).Width = 72
    End With
End Sub
```


Show All

WindowState Property

Returns or sets a **PbWindowState** constant indicating the state of the Microsoft Publisher window. Read/write.

PbWindowState can be one of these PbWindowState constants.
pbWindowStateMaximize
pbWindowStateMinimize
pbWindowStateNormal

expression.**WindowState**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

When the state of the window is **pbWindowStateNormal**, the window is neither maximized nor minimized.

Example

This example maximizes the Publisher window.

```
ActiveWindow.WindowState = pbWindowStateMaximized
```

Wizard Property

Returns a **Wizard** object representing the publication design associated with the specified publication or the wizard associated with the specified Design Gallery object.

expression.**Wizard**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

When accessing the **Wizard** property from the **Document** or **Page** object, if the specified publication is not associated with any publication design, an error occurs.

When accessing the **Wizard** property from the **Shape** or **ShapeRange** object, if the specified object is not a Design Gallery object, an error occurs.

Example

The following example reports on the publication design associated with the active publication, displaying its name and current settings.

```
Dim wizTemp As Wizard
Dim wizproTemp As WizardProperty
Dim wizproAll As WizardProperties

Set wizTemp = ActiveDocument.Wizard

With wizTemp
    Set wizproAll = .Properties
    Debug.Print "Publication design associated with " _
        & "current publication: " _
        & .Name
    For Each wizproTemp In wizproAll
        With wizproTemp
            Debug.Print "   Setting: " _
                & .Name & " = " & .CurrentValueId
        End With
    Next wizproTemp
End With
```

Note Depending on the language version of Publisher that you are using, you may receive an error when using the above code. If this occurs, you will need to build in error handlers to circumvent the errors. For more information, see **Wizard Object**.

WizardCatalogVisible Property

Returns or sets a **Boolean** indicating whether the Wizard Catalog is visible. Read/write.

expression.**WizardCatalogVisible**

expression Required. An expression that returns the **Application** object.

Example

The following example stores the current state of the Wizard Catalog in order to restore it later.

```
Sub WizardCatalogExample()
    Dim blnWizardCatalog As Boolean

    ' Store current state of Wizard Catalog.
    blnWizardCatalog = Application.WizardCatalogVisible

    ' Code can run here that shows or hides the Wizard
    ' Catalog as necessary; the original setting
    ' will be restored at the end of the procedure.

    ' Restore original state of Wizard Catalog.
    Application.WizardCatalogVisible = blnWizardCatalog
End Sub
```

WizardState Property

Returns or sets a **Long** indicating the current Mail Merge wizard step for a publication. The **WizardState** property returns a number that equates to the current Mail Merge wizard step; a zero (0) means the Mail Merge wizard is closed. Read/write.

expression.**WizardState**

expression Required. An expression that returns a **MailMerge** object.

Example

This example displays the Mail Merge wizard if it is closed.

```
Sub ShowMergeWizard()
    With ActiveDocument.MailMerge
        If .WizardState = 0 Then
            .ShowWizard
        End If
    End With
End Sub
```


Show All

WizardTag Property

Returns or sets a **PbWizardTag** constant indicating the function of a specified shape with respect to its publication design. Read/write.

PbWizardTag can be one of these PbWizardTag constants.

pbWizardTagAddress
pbWizardTagAddressGroup
pbWizardTagBriefDescriptionCaption
pbWizardTagBriefDescriptionGraphic
pbWizardTagBriefDescriptionSummary
pbWizardTagBriefDescriptionSummaryPrimary
pbWizardTagBriefDescriptionTitle
pbWizardTagBusinessDescription
pbWizardTagCustomerMailingAddress
pbWizardTagDate
pbWizardTagEAPostalCodeBox
pbWizardTagEAPostalCodeGroup
pbWizardTagEAPostalCodeLine
pbWizardTagFloatingGraphicCaption
pbWizardTagHourTimeDateInformation
pbWizardTagJobTitle
pbWizardTagLinkedStoryPrimary
pbWizardTagLinkedStorySecondary
pbWizardTagLinkedStoryTertiary
pbWizardTagList
pbWizardTagLocation
pbWizardTagLogoGroup
pbWizardTagMainFloatingGraphic
pbWizardTagMainGraphic
pbWizardTagMainTitle
pbWizardTagMapPicture

pbWizardTagMasthead
pbWizardTagNewsletterTitle
pbWizardTagOrganizationName
pbWizardTagOrganizationNameGroup
pbWizardTagPageNumber
pbWizardTagPersonalName
pbWizardTagPersonalNameGroup
pbWizardTagPhoneFaxEmail
pbWizardTagPhoneFaxEmailGroup
pbWizardTagPhoneNumber
pbWizardTagPhotePlaceholderText
pbWizardTagPhotoPlaceholderFrame
pbWizardTagPublicationDate
pbWizardTagQuickPubContent
pbWizardTagQuickPubHeading
pbWizardTagQuickPubMessage
pbWizardTagQuickPubPicture
pbWizardTagReturnAddressLines
pbWizardTagStampBox
pbWizardTagStampBoxOutline
pbWizardTagStory
pbWizardTagStoryCaptionPrimary
pbWizardTagStoryCaptionSecondary
pbWizardTagStoryGraphicPrimary
pbWizardTagStoryGraphicSecondary
pbWizardTagStoryTitle
pbWizardTagTableOfContents
pbWizardTagTableOfContentsTitle
pbWizardTagTagLine
pbWizardTagTagLineGroup
pbWizardTagTime

expression.**WizardTag**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The combination of the **WizardTagInstance** property and the **WizardTag** property uniquely defines every shape in a publication.

Example

The following example displays the wizard tag and wizard tag instance information for all the shapes on page one of the active publication.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop
        Debug.Print "Shape: " & .Name
        Debug.Print "    Wizard tag: " & .WizardTag
        Debug.Print "    Wizard tag instance: " _
            & .WizardTagInstance
    End With
Next shpLoop
```

WizardTagInstance Property

Returns or sets a **Long** indicating the instance of the specified shape compared with other shapes having the same wizard tag. Read/write.

expression.**WizardTagInstance**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

The combination of the **WizardTagInstance** property and the **WizardTag** property uniquely defines every shape in a publication.

Example

The following example displays the wizard tag and wizard tag instance information for all the shapes on page one of the active publication.

```
Dim shpLoop As Shape

For Each shpLoop In ActiveDocument.Pages(1).Shapes
    With shpLoop
        Debug.Print "Shape: " & .Name
        Debug.Print "    Wizard tag: " & .WizardTag
        Debug.Print "    Wizard tag instance: " _
            & .WizardTagInstance
    End With
Next shpLoop
```

XOffsetWithinReaderSpread Property

Returns a **Single** that represents the distance (in points) from the left edge of the reader spread to the left edge of the page. Read-only.

expression.**XOffsetWithinReaderSpread**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a shape on the second and third pages of the active publication and then sets the position of the shape on the third page to the diagonally opposite corner of the page from the shape on the second page. For this example to work, the active publication must have at least three pages.

```
Sub OffsetShapePositions()
    Dim shpOne As Shape
    Dim intLeft As Integer
    Dim intTop As Integer
    Dim intWidth As Integer
    Dim intHeight As Integer

    With ActiveDocument
        .ViewTwoPageSpread = True

        With .Pages
            intWidth = 150
            intHeight = 150
            intLeft = (.Item(2).Width / 2) - intWidth
            intTop = InchesToPoints(7)

            Set shpOne = .Item(2).Shapes.AddShape _
                (Type:=msoShape5pointStar, Left:=intLeft, _
                Top:=intTop, Width:=intWidth, Height:=intHeight)

            intLeft = (.Item(3).XOffsetWithinReaderSpread - _
                .Item(2).XOffsetWithinReaderSpread) + (.Item(2) _
                .Width - shpOne.Left - shpOne.Width)
            intTop = (.Item(3).YOffsetWithinReaderSpread - _
                .Item(2).YOffsetWithinReaderSpread) + (.Item(2) _
                .Height - shpOne.Top - shpOne.Height)

            .Item(2).Shapes.AddShape Type:=msoShape5pointStar, _
                Left:=intLeft, Top:=intTop, Width:=intWidth, _
                Height:=intHeight
        End With
    End With
End Sub
```


Show All

Yellow Property

Sets or returns a **Long** that represents the yellow component of a CMYK color. Value can be any number between 0 and 255. Read/write.

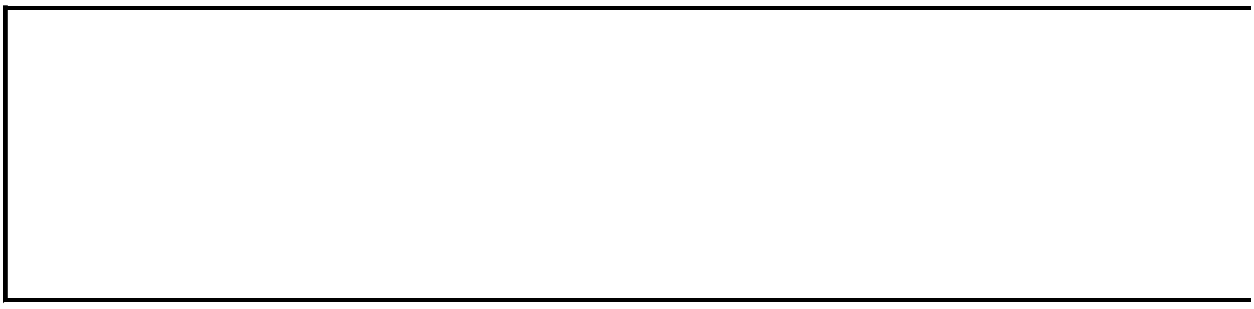
expression.**Yellow**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates two new shapes and then sets the CMYK fill color for one shape and sets the CMYK values of the second shape to the same CMYK values.

```
Sub ReturnAndSetCMYK()
    Dim lngCyan As Long
    Dim lngMagenta As Long
    Dim lngYellow As Long
    Dim lngBlack As Long
    Dim shpHeart As Shape
    Dim shpStar As Shape

    Set shpHeart = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShapeHeart, Left:=100, _
        Top:=100, Width:=100, Height:=100)
    Set shpStar = ActiveDocument.Pages(1).Shapes.AddShape _
        (Type:=msoShape5pointStar, Left:=200, _
        Top:=100, Width:=150, Height:=150)

    With shpHeart.Fill.ForeColor.CMYK
       .SetCMYK 10, 80, 200, 30
        lngCyan = .Cyan
        lngMagenta = .Magenta
        lngYellow = .Yellow
        lngBlack = .Black
    End With

    'Sets new shape to current shapes CMYK colors
    shpStar.Fill.ForeColor.CMYK.SetCMYK _
        Cyan:=lngCyan, Magenta:=lngMagenta, _
        Yellow:=lngYellow, Black:=lngBlack
End Sub
```



YOffsetWithinReaderSpread Property

Returns a **Single** that represents the distance (in points) from the top edge of the reader spread to the top edge of the page. Read-only.

expression.**YOffsetWithinReaderSpread**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

This example creates a shape on the second and third pages of the active publication and then sets the position of the shape on the third page to the diagonally opposite corner of the page from the shape on the second page. For this example to work, the active publication must have at least three pages.

```
Sub OffsetShapePositions()
    Dim shpOne As Shape
    Dim intLeft As Integer
    Dim intTop As Integer
    Dim intWidth As Integer
    Dim intHeight As Integer

    With ActiveDocument
        .ViewTwoPageSpread = True

        With .Pages
            intWidth = 150
            intHeight = 150
            intLeft = (.Item(2).Width / 2) - intWidth
            intTop = InchesToPoints(7)

            Set shpOne = .Item(2).Shapes.AddShape _
                (Type:=msoShape5pointStar, Left:=intLeft, _
                Top:=intTop, Width:=intWidth, Height:=intHeight)

            intLeft = (.Item(3).XOffsetWithinReaderSpread - _
                .Item(2).XOffsetWithinReaderSpread) + (.Item(2) _
                .Width - shpOne.Left - shpOne.Width)
            intTop = (.Item(3).YOffsetWithinReaderSpread - _
                .Item(2).YOffsetWithinReaderSpread) + (.Item(2) _
                .Height - shpOne.Top - shpOne.Height)

            .Item(2).Shapes.AddShape Type:=msoShape5pointStar, _
                Left:=intLeft, Top:=intTop, Width:=intWidth, _
                Height:=intHeight
        End With
    End With
End Sub
```


Show All

Zoom Property

Returns or sets a **PbZoom** constant or a value between 10 and 400 indicating the zoom setting of the specified view. Read/write.

PbZoom can be one of these PbZoom constants.

pbZoomFitSelection Resizes the page view to the size of the current selection.

pbZoomPageWidth Resizes the page view to the width of the publication.

pbZoomWholePage Resizes the page view to the size of a whole page.

expression.**Zoom**

expression Required. An expression that returns one of the objects in the Applies To list.

Example

The following example sets the zoom for the active publication so that the entire page fits on the screen.

```
ActiveDocument.ActiveView.Zoom = pbZoomWholePage
```

ZOrderPosition Property

Returns a **Long** indicating the position of the specified shape or shape range in the z-order. Read-only.

expression.**ZOrderPosition**

expression Required. An expression that returns one of the objects in the Applies To list.

Remarks

A shape's position in the z-order corresponds to the shape's index number in the **Shapes** collection. For example, if there are four shapes on the page, the expression `ActiveDocument.Pages(1).Shapes(1)` returns the shape at the back of the z-order, and the expression `ActiveDocument.Pages(1).Shapes(4)` returns the shape at the front of the z-order.

Whenever you add a new shape to a collection, it's added to the front of the z-order by default.

To set the shape's position in the z-order, use the **ZOrder** method.

Example

This example adds an oval to the active publication, and then places the oval second from the back in the z-order if there is at least one other shape on the page.

```
With ActiveDocument.Pages(1).Shapes _
        .AddShape(Type:=msoShapeOval, _
        Left:=100, Top:=100, Width:=100, Height:=300)
    Do While .ZOrderPosition > 2
        .ZOrder msoSendBackward
    Loop
End With
```

BeforeClose Event

Occurs immediately before any open document closes.

Private Sub Document_BeforeClose(*Cancel* As Boolean)

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the document doesn't close when the procedure is finished.

Remarks

For more information about using events with the **Document** object, see Using Events with the Document Object.

Example

This example prompts the user for a yes or no response before closing a document. For this example to work, you must place this code into the **ThisDocument** module.

```
Private Sub Document_BeforeClose(Cancel As Boolean)
    Dim intResponse As Integer

    intResponse = MsgBox("Do you really want to close " _
        & "the document?", vbYesNo)

    If intResponse = vbNo Then Cancel = True
End Sub
```

DocumentBeforeClose Event

Occurs immediately before any open document closes.

Private Sub *object*_**DocumentBeforeClose**(**ByVal** ***Doc*** **As Document**, ***Cancel*** **As Boolean**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The document that's being closed.

Cancel Optional. **False** when the event occurs. If the event procedure sets this argument to **True**, the document doesn't close when the procedure is finished.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Microsoft Publisher **Application** object, see Using Events with the Application Object.

Example

This example prompts the user for a yes or no response before closing a document. This code must be placed in a class module, and an instance of the class must be correctly initialized, using an example similar to the **SetPubApp** routine below, in order to see this example work.

```
Private WithEvents PubApp As Application

Sub SetPubApp()
    Set PubApp = Publisher.Application
End Sub

Private Sub PubApp_DocumentBeforeClose(ByVal Doc As Document, Cancel

    Dim intResponse As Integer

    intResponse = MsgBox("Do you really want to close " _
        & "the document?", vbYesNo)

    If intResponse = vbNo Then Cancel = True

End Sub
```

DocumentOpen Event

Occurs when opening a document.

Private Sub *object*_**DocumentOpen(ByVal** ***Doc*** **As Document)**

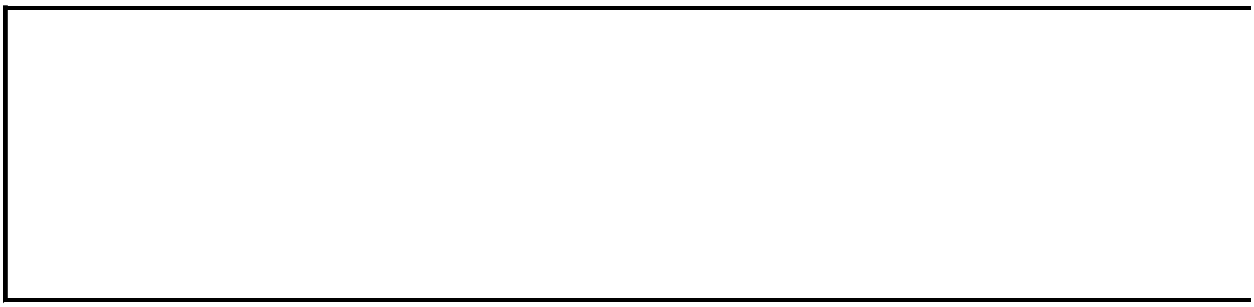
object An object of type **Application** declared with events in a class module. For more information about using events with the **Document** object, see Using Events with the Application Object.

Doc **Document**. The document that's being opened.

Example

This example displays a message with the document's name when opening a document.

```
Private Sub appPub_DocumentOpen(ByVal Doc As Document)
    MsgBox "Please wait. " & Doc.Name & " is opening."
End Sub
```



MailMergeAfterMerge Event

Occurs after all records in a mail merge have merged successfully.

Private Sub *object*_**MailMergeAfterMerge**(**ByVal** ***Doc*** **As Document**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

This example displays a message stating that all records in the specified document are finished merging.

```
Private Sub MailMergeApp_MailMergeAfterMerge(ByVal Doc As Document)

    MsgBox "Your mail merge on " & _
        ActiveDocument.Name & " is now finished."

End Sub
```

For this event to occur, you must place the following line of code in the General Declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```

MailMergeAfterRecordMerge Event

Occurs after each record in the data source successfully merges in a mail merge.

Private Sub *object*_**MailMergeAfterRecordMerge**(**ByVal** ***Doc*** **As Document**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

Remarks

If you maintain a customer management database, you can use the **MailMergeAfterRecordMerge** event to update the database for each merged record.

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

This example displays a message with the value of the first and second fields in the record that has just finished merging.

```
Private Sub MailMergeApp_MailMergeAfterRecordMerge(ByVal Doc As Docu

    With ActiveDocument.MailMerge.DataSource
        MsgBox .DataFields.Item(3).Value & " " & _
            .DataFields.Item(2).Value & " is finished merging."
    End With

End Sub
```

For this event to occur, you must place the following line of code in the General Declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```

MailMergeBeforeMerge Event

Occurs when a merge is executed before any records in a mail merge have merged.

Private Sub *object*_**MailMergeBeforeMerge**(**ByVal** ***Doc*** **As Document**, **ByVal** ***StartRecord*** **As Long**, **ByVal** ***EndRecord*** **As Long**, ***Cancel*** **As Boolean**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

StartRecord Required. The first record in the data source to include in the mail merge.

EndRecord Required. The last record in the data source to include in the mail merge.

Cancel Optional. **True** stops the mail merge process before it starts.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

This example displays a message before the mail merge process begins, asking the user if they want to continue. If the user clicks No, the merge process is cancelled.

```
Private Sub MailMergeApp_MailMergeBeforeMerge(ByVal Doc As Document,
    ByVal StartRecord As Long, ByVal EndRecord As Long, _
    Cancel As Boolean)

    Dim intVBAnswer As Integer

    Set Doc = ActiveDocument

    'Request whether the user wants to continue with the merge
    intVBAnswer = MsgBox("Mail Merge for " & Doc.Name & _
        " is now starting.  Do you want to continue?", _
        vbYesNo, "Event!")

    'If user's response to question was No, then cancel merge proces
    'and deliver a message to the user stating the merge is cancelle
    If intVBAnswer = vbNo Then
        Cancel = True
        MsgBox "You have cancelled mail merge for " & _
            Doc.Name & "."
    End If

End Sub
```

For this event to occur, you must place the following line of code in the General Declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```

MailMergeBeforeRecordMerge Event

Occurs as a merge is executed for the individual records in a merge.

Private Sub *object*_**MailMergeBeforeRecordMerge**(**ByVal** ***Doc*** **As Document**, ***Cancel*** **As Boolean**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

Cancel Optional. **True** stops the mail merge process for the current record only before it starts.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

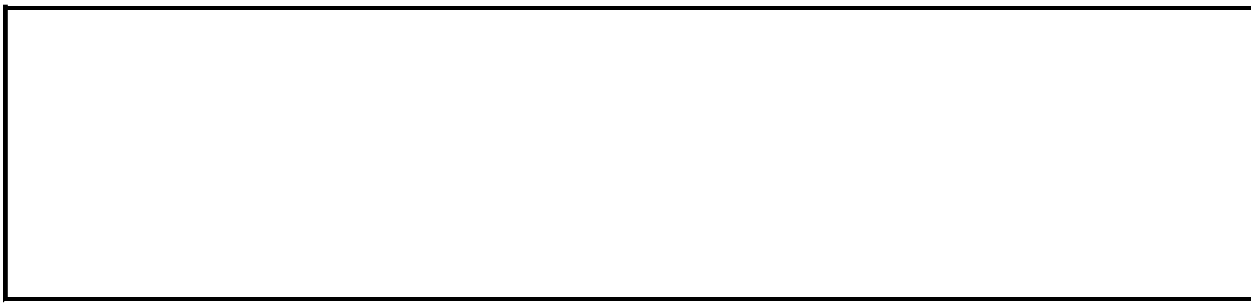
Example

This example verifies that the length of the ZIP code (which in this example is field number six) is less than five and if it is, cancels the merge for that record only.

```
Private Sub MailMergeApp_MailMergeBeforeRecordMerge(ByVal _
    Doc As Document, Cancel As Boolean)

        Dim intZipLength As Integer

        intZipLength = Len(ActiveDocument.MailMerge _
            .DataSource.DataFields(6).Value)

        'Cancel merge of this record only if
        'the ZIP code is less than five digits
        If intZipLength < 5 Then
            Cancel = True
        End If

End Sub
```

For this event to occur, you must place the following line of code in the global declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```



MailMergeDataSourceLoad Event

Occurs when the data source is loaded for a mail merge.

Private Sub *object*_**MailMergeDataSourceLoad**(**ByVal** ***Doc*** **As Document**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

This example displays a message with the data source file name when the data source starts loading.

```
Private Sub MailMergeApp_MailMergeDataSourceLoad(ByVal Doc As Docume
    Dim strDSName As String
    Dim intDSLength As Integer
    Dim intDSStart As Integer

    'Pull out of the Name property (which includes path and filename
    'only the filename using VB commands Len, InStrRev, and Right
    intDSLength = Len(ActiveDocument.MailMerge.DataSource.Name)
    intDSStart = InStrRev(ActiveDocument.MailMerge.DataSource.Name,
    intDSStart = intDSLength - intDSStart
    strDSName = Right(ActiveDocument.MailMerge.DataSource.Name, intD

    'Deliver a message to user when data source is loading
    MsgBox "Your data source, " & strDSName & ", is now loading."
End Sub
```

For this event to occur, you must place the following line of code in the General Declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```

MailMergeDataSourceValidate Event

Occurs when a user performs address verification by clicking **Validate** in the **Mail Merge Recipients** dialog box.

Private Sub *object*_**MailMergeDataSourceValidate**(**ByVal** ***Doc*** **As Document**, ***Handled*** **As Boolean**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

Handled Optional. **True** runs the accompanying validation code against the mail merge data source. **False** cancels the data source validation.

Remarks

If you don't have address verification software installed on your computer, use the **MailMergeDataSourceValidate** event to create simple filtering routines, such as looping through records to check the postal codes and remove any that are non-U.S. Non-U.S. users can filter out all U.S. postal codes by modifying the code sample below and using Visual Basic commands to search for text or special characters.

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

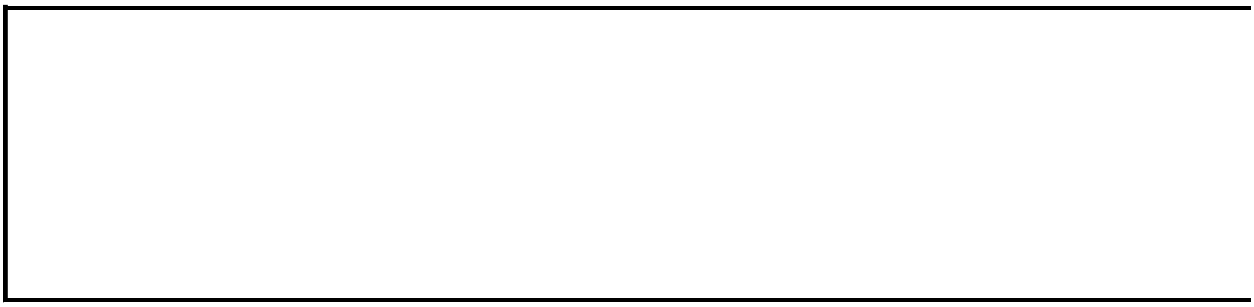
This example validates ZIP codes in the attached data source for five digits. If the length of the ZIP code is less than five, the record is excluded from the mail merge process. This example assumes the postal codes are U.S. ZIP codes. You could modify this example to search for ZIP codes that have a 4-digit locator code appended to the ZIP code, and then exclude all records that don't contain the locator code.

```
Private Sub MailMergeApp_MailMergeDataSourceValidate( _
        ByVal Doc As Document, _
        Handled As Boolean)

    Dim intCount As Integer

    Handled = True

    On Error Resume Next

    With ActiveDocument.MailMerge.DataSource

        'Set the active record equal to the first included record in
        'data source
        .ActiveRecord = 1
        Do
            intCount = intCount + 1

            'Set the condition that field six must be greater than o
            'equal to five
            If Len(.DataFields.Item(6).Value) < 5 Then

                'Exclude the record if field six is less than five d
                .Included = False

                'Mark the record as containing an invalid address fi
                .InvalidAddress = True

                'Specify the comment attached to the record explaini
                'why the record was excluded from the mail merge
                .InvalidComments = "The ZIP code for this record is
                    & "less than five digits. It will be removed " _
                    & "from the mail merge process."

            End If
```

```
            'Move the record to the next record in the data source
            .ActiveRecord = .ActiveRecord + 1

        'End the loop when the counter variable
        'equals the number of records in the data source
        Loop Until intCount = .RecordCount
    End With

End Sub
```

For this event to occur, you must place the following line of code in the General Declarations section of your module and run the following initialization routine.

```
Private WithEvents MailMergeApp As Application

Sub InitializeMailMergeApp()
    Set MailMergeApp = Publisher.Application
End Sub
```



MailMergeWizardStateChange Event

Occurs when a user changes from a specified step to a specified step in the Mail Merge Wizard.

Private Sub *object*_**MailMergeWizardStateChange**(**ByVal** ***Doc*** **As Document**, ***FromState*** **As Long**)

object A variable which references an object of type **Application** declared with events in a class module.

Doc Required. The mail merge main document.

FromState Optional. The Mail Merge Wizard step from which a user is moving.

Remarks

To access the **Application** object events, declare an **Application** object variable in the General Declarations section of a code module. Then set the variable equal to the **Application** object for which you want to access events. For information about using events with the Publisher **Application** object, see Using Events with the Application Object.

Example

This example displays a message when a users moves from step three of the Mail Merge Wizard to step four. Based on the user's answer to the message, the user will either continue on to step four or return to step three.

```
Private Sub MailMergeApp_MailMergeWizardStateChange(ByVal Doc As Doc
        ByVal FromState As Long)

   Select Case FromState
        Case 1
            MsgBox "Now you will build your publication merge " & _
                "by adding fields to your publication."
        Case 2
            MsgBox "Now you will see your publication " & _
                "merged with the records in the data source."
        Case 3
            MsgBox "Now you will complete the mail merge process."
    End Select

End Sub
```

NewDocument Event

Occurs when a new publication is created.

Private Sub *object*_**NewDocument(ByVal** ***Doc*** **As Document)**

object An object of type **Application** declared with events in a class module. For more information about using events with the **Application** object, see Using Events with the Application Object.

Doc The new document.

Example

This example displays a message when a new publication is created.

```
Private Sub appPub_NewDocument(ByVal Doc As Document)
    MsgBox "This is a new publication."
End Sub
```

Open Event

Occurs when a publication is opening.

Private Sub *object*_**Open**( )

object A variable which references an object of type **Document** declared with events in a class module.

Remarks

To access the **Document** object events, declare a **Document** object variable in the General Declarations section of a class module, then set the variable equal to the **Document** object for which you want to access events.

For more information about using events with the **Document** object, see Using Events with the Document Object.

Example

This example displays a message when a publication is opened. (The procedure can be stored in the **ThisDocument** module of a publication.)

```
Private Sub Document_Open()
    MsgBox "This publication is copyrighted."
End Sub
```

Quit Event

Occurs when the user quits Microsoft Publisher.

Private Sub *object*_**Quit**( )

object A variable which references an object of type **Application** declared with events in a class module.

Remarks

For information about using events with the Application object, see Using Events with the Application Object.

Example

This example ensures that the **Standard** and **Formatting** toolbars are visible before the user quits Publisher. As a result, when Publisher is started again, the **Standard** and **Formatting** toolbars will be visible.

This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see Using Events with the Application Object for directions on how to accomplish this.

```
Public WithEvents appPublisher as Publisher.Application

Private Sub appPublisher_Quit()
    CommandBars("Standard").Visible = True
    CommandBars("Formatting").Visible = True
End Sub
```

Redo Event

Occurs when reversing the last action that was undone.

Private Sub *object*_**Redo**( )

object Required. A variable which references an object of type **Document** declared with events in a class module.

Remarks

The **Redo** event occurs immediately after the action is redone.

If multiple actions are redone, the **Redo** event only occurs once after all the actions are complete.

For more information about using events with the **Document** object, see Using Events with the Document Object.

Example

This example displays a message when a user clicks the **Redo** button on the **Standard** toolbar or selects **Redo** from the **Edit** menu. For this routine to work with the current publication, you must put it in the ThisDocument module.

```
Private Sub DocPub_Redo()
    MsgBox "Your last undo has been reversed."
End Sub
```

To trap this event from a non-Publisher project, you must place the following code in the General Declarations section of your module and run the InitiatePubApp routine.

```
Private WithEvents DocPub As Publisher.Document

Sub InitiatePubApp()
    Set DocPub = Publisher.ActiveDocument
End Sub
```

ShapesAdded Event

Occurs when one or more new shapes are added to a publication. This event occurs whether shapes are added manually or programmatically.

Private Sub Document_ShapesAdded()

Example

This example displays a message whenever a new shape is added to the active publication. For this example to work, you must place this code into the **ThisDocument** module.

```
Private Sub PubDoc_ShapesAdded()
    MsgBox "You just added a new shape."
End Sub
```

ShapesRemoved Event

Occurs when a shape is deleted from a publication.

Private Sub Document_ShapesRemoved()

Example

This example displays a message whenever a shape is removed from the active publication. For this example to work, you must place this code into the **ThisDocument** module.

```
Private Sub Document_ShapesRemoved()
    MsgBox "You just deleted one or more shapes."
End Sub
```

Undo Event

Occurs when a user undoes the last action performed.

Private Sub *object*_**Undo**( )

object Required. A variable which references an object of type **Document** declared with events in a class module.

Remarks

The **Undo** event occurs immediately after the action is undone.

If multiple actions are undone, the **Undo** event only occurs once after all the actions are undone.

For more information about using events with the **Document** object, see Using Events with the Document Object.

Example

This example displays a message when the user clicks on the **Undo** button on the **Standard** toolbar or selects **Undo** from the **Edit** menu. For this routine to work with the current publication, you must put it in the ThisDocument module.

```
Private Sub DocPub_Undo()
    MsgBox "Your last action has been reversed."
End Sub
```

To trap this event from a non-Publisher project, you must place the following code in the General Declarations section of your module and run the InitiatePubApp routine.

```
Private WithEvents DocPub As Publisher.Document

Sub InitiatePubApp()
    Set DocPub = Publisher.ActiveDocument
End Sub
```

WindowActivate Event

Occurs when the application window is activated.

Private Sub *object*_**WindowActivate**(**ByVal** ***Wn*** **As Window**)

object A variable which references an object of type **Application** declared with events in a class module.

Wn Required. The window that's being activated.

Remarks

For information about using events with the Application object, see Using Events with the Application Object.

Example

This example maximizes the Microsoft Publisher window when it's activated. This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see Using Events with the Application Object for directions on how to accomplish this.

```
Public WithEvents appPublisher as Publisher.Application

Private Sub appPublisher_WindowActivate _
        (ByVal Wn As Window)
    Wn.WindowState = pbWindowStateMaximize
End Sub
```

WindowDeactivate Event

Occurs when the application window is deactivated.

Private Sub *object*_**WindowDeactivate**(**ByVal** ***Wn*** **As Window**)

object A variable which references an object of type **Application** declared with events in a class module.

Wn Required. The window that's being deactivated.

Remarks

For information about using events with the Application object, see Using Events with the Application Object.

Example

This example minimizes the window when it's deactivated. This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see Using Events with the Application Object for directions on how to accomplish this.

```
Public WithEvents appPublisher as Publisher.Application

Private Sub appPublisher_WindowDeactivate _
        (ByVal Wn As Window)
    Wn.WindowState = pbWindowStateMinimize
End Sub
```

WindowPageChange Event

Occurs when switching the view from one page to another page in a publication.

Private Sub *object*_**WindowPageChange(ByVal** *Vw* **As View)**

object An object of type **Application** declared with events in a class module. For more information about using events with the **Application** object, see Using Events with the Application Object.

vw The new view that includes the page to which the view has been switched.

Example

This example changes the view to display the whole page when switching to a new page in a publication. For this example to work, you must place the **WithEvents** declaration in the General Declarations section of a class module and run the InitializeEvents routine.

```
Private WithEvents PubApp As Publisher.Application

Sub InitializeEvents()
    Set PubApp = Publisher.Application
End Sub

Private Sub PubApp_WindowPageChange(ByVal Vw As View)
    Vw.Zoom = pbZoomWholePage
End Sub
```

WizardAfterChange Event

Occurs after the user chooses an option in the wizard pane that changes any of the following settings in the publication: page layout (page size, fold type, orientation, label product), print setup (paper size, print tiling), adding or deleting objects, adding or deleting pages, or object or page formatting (size, position, fill, border, background, default text, text formatting).

Private Sub *object*_**WizardAfterChange**( )

object A variable which references an object of type **Document** declared with events in a class module.

Remarks

The WizardAfterChange event only occurs once regardless of the scope or number of individual modifications made to the publication.

To access the **Document** object events, declare a **Document** object variable in the General Declarations section of a class module, then set the variable equal to the **Document** object for which you want to access events.

For more information about using events with the **Document** object, see Using Events with the Document Object.

Example

This example displays a message when a publication is altered using the wizard pane. (The procedure can be stored in the ThisDocument module of a publication.)

```
Private Sub Document_WizardAfterChange()
    MsgBox "Remember to save changes made " _
        & "through the wizard pane!"
End Sub
```


Show All

Publisher Constants

This topic provides a list of all enumerated constants in the Publisher object model.

PbCalendarType

Constant	Value
pbCalendarTypeArabicHijri	1
pbCalendarTypeChineseNational	3
pbCalendarTypeHebrewLunar	2
pbCalendarTypeJapaneseEmperor	4
pbCalendarTypeKoreanDanki	6
pbCalendarTypeSakaEra	7
pbCalendarTypeThaiBuddhist	5
pbCalendarTypeTranslitEnglish	8
pbCalendarTypeTranslitFrench	9
pbCalendarTypeWestern	0

PbCellDiagonalType

Constant	Value
pbTableCellDiagonalDown	2
pbTableCellDiagonalMixed	-2
pbTableCellDiagonalNone	0
pbTableCellDiagonalUp	1

PbCollapseDirection

Constant	Value
pbCollapseEnd	2
pbCollapseStart	1

PbColorMode

Constant	Value
pbColorModeBW	3
pbColorModeDesktop	0
pbColorModeProcess	1
pbColorModeSpot	2
pbColorModeSpotAndProcess	4

PbColorModel

Constant	Value
pbColorModelCMYK	2
pbColorModelGreyScale	3
pbColorModelRGB	1
pbColorModelUnknown	4

PbColorScheme

Constant	Value
pbColorSchemeAlpine	-1
pbColorSchemeAqua	-2
pbColorSchemeBerry	-3
pbColorSchemeBlackGray	-4
pbColorSchemeBlackWhite	-58
pbColorSchemeBrown	-5
pbColorSchemeBurgundy	-6
pbColorSchemeCavern	-7
pbColorSchemeCelebration	-1004
pbColorSchemeCherry	-1002
pbColorSchemeCitrus	-8
pbColorSchemeClay	-9
pbColorSchemeCranberry	-10
pbColorSchemeCrocus	-11

pbColorSchemeCustom	1
pbColorSchemeDarkBlue	-61
pbColorSchemeDesert	-12
pbColorSchemeField	-13
pbColorSchemeFirstUserDefined	2000
pbColorSchemeFjord	-14
pbColorSchemeFloral	-15
pbColorSchemeGarnet	-16
pbColorSchemeGlacier	-17
pbColorSchemeGreen	-59
pbColorSchemeHeather	-18
pbColorSchemeIris	-19
pbColorSchemeIsland	-20
pbColorSchemeIvy	-21
pbColorSchemeLagoon	-22
pbColorSchemeLilac	-23
pbColorSchemeMahogany	-24
pbColorSchemeMarine	-25
pbColorSchemeMaroon	-26
pbColorSchemeMeadow	-27
pbColorSchemeMist	-28
pbColorSchemeMistletoe	-29
pbColorSchemeMonarch	-41
pbColorSchemeMoss	-30
pbColorSchemeMountain	-31
pbColorSchemeMulberry	-32
pbColorSchemeNavy	-33
pbColorSchemeNutmeg	-34
pbColorSchemeOcean	-1000
pbColorSchemeOlive	-35
pbColorSchemeOrange	-1003
pbColorSchemeOrchid	-36
pbColorSchemeParrot	-37

pbColorSchemePeach	-1005
pbColorSchemePebbles	-38
pbColorSchemePrairie	-39
pbColorSchemePurple	-1001
pbColorSchemeRainForest	-40
pbColorSchemeRed	-60
pbColorSchemeRedwood	-42
pbColorSchemeReef	-43
pbColorSchemeSagebrush	-44
pbColorSchemeSapphire	-45
pbColorSchemeShamrock	-46
pbColorSchemeSienna	-47
pbColorSchemeSpice	-48
pbColorSchemeSunrise	-49
pbColorSchemeSunset	-50
pbColorSchemeTeal	-51
pbColorSchemeTidepool	-52
pbColorSchemeTropics	-53
pbColorSchemeTrout	-54
pbColorSchemeVineyard	-55
pbColorSchemeWaterfall	-56
pbColorSchemeWildflower	-57

PbColorType

Constant	Value
pbColorTypeCMS	4
pbColorTypeCMYK	3
pbColorTypeInk	5
pbColorTypeMixed	-2
pbColorTypeRGB	1
pbColorTypeScheme	2

PbCommandButtonType

Constant	Value
pbCommandButtonReset	2
pbCommandButtonSubmit	1

PbDateTimeFormat

Constant	Value
pbDateEnglish	8
pbDateISO	4
pbDateLong	2
pbDateLongDay	1
pbDateMon_Yr	10
pbDateMonthYr	9
pbDateShort	0
pbDateShortAbb	7
pbDateShortAlt	3
pbDateShortMon	5
pbDateShortSlash	6
pbDateTimeEastAsia1	17
pbDateTimeEastAsia2	18
pbDateTimeEastAsia3	19
pbDateTimeEastAsia4	20
pbDateTimeEastAsia5	21
pbTime24	15
pbTimeDatePM	11
pbTimeDateSecPM	12
pbTimePM	13
pbTimeSec24	16
pbTimeSecPM	14

PbDirectionType

Constant	Value
pbDirectionLeftToRight	1
pbDirectionRightToLeft	2

PbFieldType

Constant	Value
pbFieldDateTime	4
pbFieldHyperlinkAbsolutePage	11
pbFieldHyperlinkEmail	12
pbFieldHyperlinkFile	13
pbFieldHyperlinkRelativePage	10
pbFieldHyperlinkURL	9
pbFieldIHIV	6
pbFieldMailMerge	5
pbFieldNone	0
pbFieldPageNumber	1
pbFieldPageNumberNext	2
pbFieldPageNumberPrev	3
pbFieldPhoneticGuide	7
pbFieldWizardSampleText	8

PbFileFormat

Constant	Value
pbFileHTMLFiltered	7
pbFilePublication	1
pbFilePublicationHTML	4
pbFilePublisher2000	3
pbFilePublisher98	2
pbFileRTF	6
pbFileWebArchive	5

PbFilterComparison

Constant	Value
pbComparisonEqual	0
pbComparisonGreaterThan	3
pbComparisonGreaterThanEqual	5
pbComparisonIsBlank	6
pbComparisonIsNotBlank	7
pbComparisonLessThan	2
pbComparisonLessThanEqual	4
pbComparisonNotEqual	1

PbFilterConjunction

Constant	Value
pbConjunctionAnd	0
pbConjunctionOr	1

PbFontLicenseLimitations

Constant	Value
pbFontEmbeddable	1
pbFontNotEmbeddable	3
pbFontPrintPreviewEmbeddable	2

PbFontScriptType

Constant	Value
pbFontScriptArabic	7
pbFontScriptArmenian	5
pbFontScriptAsciiLatin	1
pbFontScriptAsciiSym	43
pbFontScriptBengali	9
pbFontScriptBopomofo	23
pbFontScriptBraille	41

pbFontScriptCanadianAbor	36
pbFontScriptCherokee	35
pbFontScriptCurrency	42
pbFontScriptCyrillic	4
pbFontScriptDefault	0
pbFontScriptDevanagari	8
pbFontScriptEthiopic	34
pbFontScriptEUDC	26
pbFontScriptGeorgian	20
pbFontScriptGreek	3
pbFontScriptGujarati	11
pbFontScriptGurmukhi	10
pbFontScriptHalfWidthKana	25
pbFontScriptHan	24
pbFontScriptHangul	21
pbFontScriptHanSurrogate	28
pbFontScriptHebrew	6
pbFontScriptKana	22
pbFontScriptKannada	15
pbFontScriptKhmer	39
pbFontScriptLao	18
pbFontScriptLatin	2
pbFontScriptMalayalam	16
pbFontScriptMixed	-2
pbFontScriptMongolian	40
pbFontScriptMyanmar	32
pbFontScriptNonHanSurrogate	29
pbFontScriptOgham	37
pbFontScriptOriya	12
pbFontScriptRunic	38
pbFontScriptSinhala	33
pbFontScriptSyriac	30
pbFontScriptTamil	13

pbFontScriptTelugu	14
pbFontScriptThaana	31
pbFontScriptThai	17
pbFontScriptTibetan	19
pbFontScriptYi	27

PbFontSource

Constant	Value
pbFontDocument	2
pbFontMissing	3
pbFontSystem	1

PbFontType

Constant	Value
pbFontPrinter	2
pbFontRaster	3
pbFontTrueType	1
pbFontUnknown	4

PbHelpType

Constant	Value
pbHelp	1
pbHelpActiveWindow	2
pbHelpPSSHelp	3

PbHlinkTargetType

Constant	Value
pbHlinkTargetTypeEmail	2
pbHlinkTargetTypeFirstPage	3
pbHlinkTargetTypeLastPage	4

pbHlinkTargetTypeNextPage	5
pbHlinkTargetTypeNone	0
pbHlinkTargetTypePageID	7
pbHlinkTargetTypePreviousPage	6
pbHlinkTargetTypeURL	1

PbHorizontalPictureLocking

Constant	Value
pbHorizontalLockingLeft	1
pbHorizontalLockingNone	0
pbHorizontalLockingRight	2
pbHorizontalLockingStretch	3

PbImageFormat

Constant	Value
pbImageFormatCMYKJPEG	10
pbImageFormatDIB	7
pbImageFormatEMF	2
pbImageFormatGIF	8
pbImageFormatJPEG	5
pbImageFormatPICT	4
pbImageFormatPNG	6
pbImageFormatTIFF	9
pbImageFormatUNKNOWN	1
pbImageFormatWMF	3

PbInkName

Constant	Value
pbInkNameBlack	4
pbInkNameCyan	1
pbInkNameMagenta	2

pbInkNameSpot1	257
pbInkNameSpot10	266
pbInkNameSpot11	267
pbInkNameSpot12	268
pbInkNameSpot2	258
pbInkNameSpot3	259
pbInkNameSpot4	260
pbInkNameSpot5	261
pbInkNameSpot6	262
pbInkNameSpot7	263
pbInkNameSpot8	264
pbInkNameSpot9	265
pbInkNameYellow	3

PbInksToPrint

Constant	Value
pbInksToPrintAll	1
pbInksToPrintConvertSpotToProcess	3
pbInkstoPrintUsed	2

PbInlineAlignment

Constant	Value
pbInlineAlignmentCharacter	0
pbInlineAlignmentLeft	1
pbInlineAlignmentMixed	-2
pbInlineAlignmentRight	2

PbLineSpacingRule

Constant	Value
pbLineSpacing1pt5	1
pbLineSpacingDouble	2

pbLineSpacingExactly	4
pbLineSpacingMixed	-9999999
pbLineSpacingMultiple	5
pbLineSpacingSingle	0

PbLinkedFileStatus

Constant	Value
pbLinkedFileMissing	2
pbLinkedFileModified	3
pbLinkedFileOK	1

PbListSeparator

Constant	Value
pbListSeparatorColon	327680
pbListSeparatorDoubleHyphen	458752
pbListSeparatorDoubleParen	65536
pbListSeparatorDoubleSquare	393216
pbListSeparatorParenthesis	0
pbListSeparatorPeriod	131072
pbListSeparatorPlain	196608
pbListSeparatorSquare	262144
pbListSeparatorWideComma	524288

PbListType

Constant	Value
pbListTypeAiueo	12
pbListTypeArabic	0
pbListTypeArabic1	46
pbListTypeArabic2	48
pbListTypeArabicLeadingZero	22
pbListTypeBullet	23

pbListTypeCardinalText	6
pbListTypeChnDbNum2	38
pbListTypeChnDbNum3	39
pbListTypeChosung	25
pbListTypeCirclenum	18
pbListTypeDAiueo	20
pbListTypeDbChar	14
pbListTypeDbNum1	10
pbListTypeDbNum2	11
pbListTypeDbNum3	16
pbListTypeDbNum4	17
pbListTypeDIroha	21
pbListTypeGanada	24
pbListTypeHebrew1	45
pbListTypeHebrew2	47
pbListTypeHindi1	49
pbListTypeHindi2	50
pbListTypeHindi3	51
pbListTypeHindi4	52
pbListTypeIroha	13
pbListTypeKorDbNum1	41
pbListTypeKorDbNum2	42
pbListTypeKorDbNum3	43
pbListTypeKorDbNum4	44
pbListTypeLowerCaseLetter	4
pbListTypeLowerCaseRoman	2
pbListTypeLowerCaseRussian	58
pbListTypeNone	255
pbListTypeOrdinal	5
pbListTypeOrdinalText	7
pbListTypeThai1	53
pbListTypeThai2	54
pbListTypeThai3	55

pbListTypeTpeDbNum2	34
pbListTypeTpeDbNum3	35
pbListTypeUpperCaseLetter	3
pbListTypeUpperCaseRoman	1
pbListTypeUpperCaseRussian	59
pbListTypeVietnamese1	56
pbListTypeZodiac1	30
pbListTypeZodiac2	31

PbMailMergeDataFieldType

Constant	Value
pbMailMergeDataFieldPicture	1
pbMailMergeDataFieldString	0

PbMailMergeDataSource

Constant	Value
pbMergeInfoFromODSO	5

PbMailMergeDestination

Constant	Value
pbMergeToExistingPublication	3
pbMergeToNewPublication	2
pbSendToPrinter	1

PbMappedDataFields

Constant	Value
pbAddress1	10
pbAddress2	11
pbAddress3	29
pbBusinessFax	17

pbBusinessPhone	16
pbCity	12
pbCompany	9
pbCountryRegion	15
pbCourtesyTitle	2
pbDepartment	30
pbEmailAddress	20
pbFirstName	3
pbHomeFax	19
pbHomePhone	18
pbJobTitle	8
pbLastName	5
pbMiddleName	4
pbNickname	7
pbPostalCode	14
pbRubyFirstName	27
pbRubyLastName	28
pbSpouseCourtesyTitle	22
pbSpouseFirstName	23
pbSpouseLastName	25
pbSpouseMiddleName	24
pbSpouseNickname	26
pbState	13
pbSuffix	6
pbUniqueIdentifier	1
pbWebPageURL	21

PbMasterPageType

Constant	Value
pbMasterPageLeftPage	1
pbMasterPageRightPage	2

PbNavBarOrientation

Constant	Value
pbNavBarOrientHorizontal	1
pbNavBarOrientVertical	2

PbOrientationType

Constant	Value
pbOrientationLandscape	2
pbOrientationPortrait	1

PbOriginalFormat

Constant	Value
pbOriginalPublicationFormat	1
pbPublisherFile	2

PbPageNumberFormat

Constant	Value
pbPageNumberFormatAiueo	12
pbPageNumberFormatArabic	0
pbPageNumberFormatArabic1	46
pbPageNumberFormatArabic2	48
pbPageNumberFormatArabicLZ	22
pbPageNumberFormatCardtext	6
pbPageNumberFormatChnDbNum2	38
pbPageNumberFormatChnDbNum3	39
pbPageNumberFormatChosung	25
pbPageNumberFormatCirclenum	18
pbPageNumberFormatDAiueo	20
pbPageNumberFormatDbChar	14
pbPageNumberFormatDbNum1	10

pbPageNumberFormatDbNum2	11
pbPageNumberFormatDbNum3	16
pbPageNumberFormatDIroha	21
pbPageNumberFormatGanada	24
pbPageNumberFormatHebrew1	45
pbPageNumberFormatHebrew2	47
pbPageNumberFormatHindi1	49
pbPageNumberFormatHindi2	50
pbPageNumberFormatHindi3	51
pbPageNumberFormatHindi4	52
pbPageNumberFormatIroha	13
pbPageNumberFormatKorDbNum1	41
pbPageNumberFormatKorDbNum2	42
pbPageNumberFormatKorDbNum3	43
pbPageNumberFormatKorDbNum4	44
pbPageNumberFormatLCLetter	4
pbPageNumberFormatLCRoman	2
pbPageNumberFormatLCRus	58
pbPageNumberFormatOrdinal	5
pbPageNumberFormatOrdtext	7
pbPageNumberFormatThai1	53
pbPageNumberFormatThai2	54
pbPageNumberFormatThai3	55
pbPageNumberFormatTpeDbNum2	34
pbPageNumberFormatTpeDbNum3	35
pbPageNumberFormatUCLetter	3
pbPageNumberFormatUCRoman	1
pbPageNumberFormatUCRus	59
pbPageNumberFormatViet1	56
pbPageNumberFormatZodiac1	30
pbPageNumberFormatZodiac2	31

PbPageNumberType

Constant	Value
pbPageNumberCurrent	1
pbPageNumberNextInStory	2
pbPageNumberPreviousInStory	3

PbPageType

Constant	Value
pbPageLeftPage	1
pbPageMasterPage	4
pbPageRightPage	2
pbPageScratchPage	3

PbParagraphAlignmentType

Constant	Value
pbParagraphAlignmentCenter	1
pbParagraphAlignmentDistribute	4
pbParagraphAlignmentDistributeAll	9
pbParagraphAlignmentDistributeCenterLast	10
pbParagraphAlignmentDistributeEastAsia	5
pbParagraphAlignmentInterCluster	8
pbParagraphAlignmentInterIdeograph	7
pbParagraphAlignmentInterWord	3
pbParagraphAlignmentJustified	6
pbParagraphAlignmentKashida	11
pbParagraphAlignmentLeft	0
pbParagraphAlignmentMixed	-9999999
pbParagraphAlignmentRight	2

PbPersonalInfoSet

Constant	Value
pbPersonalInfoHome	4

pbPersonalInfoOtherOrganization	3
pbPersonalInfoPrimaryBusiness	1
pbPersonalInfoSecondaryBusiness	2

PbPhoneticGuideAlignmentType

Constant	Value
pbPhoneticGuideAlignmentCenter	3
pbPhoneticGuideAlignmentDefault	0
pbPhoneticGuideAlignmentLeft	4
pbPhoneticGuideAlignmentOneTwoOne	2
pbPhoneticGuideAlignmentRight	5
pbPhoneticGuideAlignmentZeroOneZero	1

PbPictureInsertAs

Constant	Value
pbPictureInsertAsEmbedded	1
pbPictureInsertAsLinked	2
pbPictureInsertAsOriginalState	3

PbPictureResolution

Constant	Value
pbPictureResolutionCommercialPrint_300dpi	3
pbPictureResolutionDefault	0
pbPictureResolutionDesktopPrint_150dpi	2
pbPictureResolutionWeb_96dpi	1

PbPlacementType

Constant	Value
pbPlacementCenter	3
pbPlacementLeft	1

pbPlacementRight	2

PbPrintGraphics

Constant	Value
pbPrintHighResolution	1
pbPrintLowResolution	2
pbPrintNoGraphics	3

PbPrintMode

Constant	Value
pbPrintModeCompositeCMYK	3
pbPrintModeCompositeGrayscale	4
pbPrintModeCompositeRGB	1
pbPrintModeSeparations	2

PbPublicationLayout

Constant	Value
pbLayout4x6BaePan	10
pbLayout4x6BanPan	12
pbLayout4x6Pan	11
pbLayoutBannerCustom	27
pbLayoutBannerLarge	26
pbLayoutBannerMedium	25
pbLayoutBannerSmall	24
pbLayoutBook	2
pbLayoutBusinessCardEurope	18
pbLayoutBusinessCardFE	19
pbLayoutBusinessCardLocal	20
pbLayoutBusinessCardUS	17
pbLayoutCrownPan	13
pbLayoutCustom	23

pbLayoutEnvelope	33
pbLayoutFoldCard	3
pbLayoutFullPage	1
pbLayoutGreetingCardL	4
pbLayoutGreetingCardT	5
pbLayoutIndexCard	16
pbLayoutJang4x6Pan	15
pbLayoutKookBaePan	6
pbLayoutKookBanPan	9
pbLayoutKookPan	7
pbLayoutLabel	32
pbLayoutPostcardA4	30
pbLayoutPostcardHalfLetter	29
pbLayoutPostcardJapan	31
pbLayoutPostcardUS	28
pbLayoutPosterLarge	22
pbLayoutPosterSmall	21
pbLayoutShinKookPan	8
pbLayoutShinSeoPan	14
pbLayoutWebPageLarge	35
pbLayoutWebPageSmall	34

PbPublicationType

Constant	Value
pbTypePrint	1
pbTypeWeb	2

PbReplaceScope

Constant	Value
pbReplaceScopeAll	2
pbReplaceScopeNone	0
pbReplaceScopeOne	1

PbReplaceTint

Constant	Value
pbReplaceTintKeepTints	1
pbReplaceTintMaintainLuminosity	2
pbReplaceTintUseDefault	0

PbRulerGuideType

Constant	Value
pbRulerGuideTypeHorizontal	2
pbRulerGuideTypeVertical	1

PbSaveOptions

Constant	Value
pbDoNotSaveChanges	3
pbPromptToSaveChanges	1
pbSaveChanges	2

PbSchemeColorIndex

Constant	Value
pbSchemeColorAccent1	2
pbSchemeColorAccent2	3
pbSchemeColorAccent3	4
pbSchemeColorAccent4	5
pbSchemeColorAccent5	8
pbSchemeColorFollowedHyperlink	7
pbSchemeColorHyperlink	6
pbSchemeColorMain	1
pbSchemeColorNone	0

PbSelectionType

Constant	Value
pbSelectionNone	0
pbSelectionShape	1
pbSelectionShapeSubSelection	4
pbSelectionTableCells	3
pbSelectionText	2

PbShapeType

Constant	Value
pbAutoShape	1
pbCallout	2
pbCatalogMergeArea	111
pbChart	3
pbComment	4
pbEmbeddedOLEObject	7
pbFormControl	8
pbFreeform	5
pbGroup	6
pbGroupWizard	108
pbLine	9
pbLinkedOLEObject	10
pbLinkedPicture	11
pbMedia	16
pbOLEControlObject	12
pbPicture	13
pbPlaceholder	14
pbShapeTypeMixed	-2
pbTable	18
pbTextEffect	15
pbTextFrame	17

pbWebCheckBox	100
pbWebCommandButton	101
pbWebHotSpot	110
pbWebHTMLFragment	107
pbWebListBox	102
pbWebMultiLineTextBox	103
pbWebNavigationBar	112
pbWebOptionButton	104
pbWebSingleLineTextBox	105
pbWebWebComponent	106

PbShowDialog

Constant	Value
pbDefaultBehavior	1
PbShowDialog	2
pbSuppressDialog	3

PbSpotColor

Constant	Value
pbInkNone	0

PbStoryType

Constant	Value
pbStoryContinuedFrom	2
pbStoryContinuedOn	3
pbStoryTable	0
pbStoryTextFrame	1

PbSubmitDataFormatType

Constant	Value

pbSubmitDataFormatCSV	3
pbSubmitDataFormatHTML	1
pbSubmitDataFormatRichText	2
pbSubmitDataFormatTab	4

PbSubmitDataRetrievalMethodType

Constant	Value
pbSubmitDataRetrievalEmail	2
pbSubmitDataRetrievalProgram	3
pbSubmitDataRetrievalSaveOnServer	1

PbTabAlignmentType

Constant	Value
pbTabAlignmentCenter	1
pbTabAlignmentDecimal	3
pbTabAlignmentLeading	0
pbTabAlignmentTrailing	2

PbTabLeaderType

Constant	Value
pbTabLeaderBullet	5
pbTabLeaderDashes	2
pbTabLeaderDot	1
pbTabLeaderLine	3
pbTabLeaderNone	0

PbTableAutoFormatType

Constant	Value
pbTableAutoFormatCheckbookRegister	0
pbTableAutoFormatCheckerboard	20

pbTableAutoFormatDefault	-3
pbTableAutoFormatList1	1
pbTableAutoFormatList2	2
pbTableAutoFormatList3	3
pbTableAutoFormatList4	4
pbTableAutoFormatList5	5
pbTableAutoFormatList6	6
pbTableAutoFormatList7	7
pbTableAutoFormatListWithTitle1	8
pbTableAutoFormatListWithTitle2	9
pbTableAutoFormatListWithTitle3	10
pbTableAutoFormatMixed	-1
pbTableAutoFormatNone	-2
pbTableAutoFormatNumbers1	11
pbTableAutoFormatNumbers2	12
pbTableAutoFormatNumbers3	13
pbTableAutoFormatNumbers4	14
pbTableAutoFormatNumbers5	15
pbTableAutoFormatNumbers6	16
pbTableAutoFormatTableOfContents1	17
pbTableAutoFormatTableOfContents2	18
pbTableAutoFormatTableOfContents3	19

PbTableDirectionType

Constant	Value
pbTableDirectionLeftToRight	1
pbTableDirectionRightToLeft	2

PbTextAutoFitType

Constant	Value
pbTextAutoFitBestFit	2
pbTextAutoFitNone	0

pbTextAutoFitShrinkOnOverflow 1

PbTextDirection

Constant	Value
pbTextDirectionLeftToRight	1
pbTextDirectionMixed	-9999999
pbTextDirectionRightToLeft	2

PbTextOrientation

Constant	Value
pbTextOrientationHorizontal	1
pbTextOrientationMixed	-2
pbTextOrientationRightToLeft	256
pbTextOrientationVerticalEastAsia	2

PbTextUnit

Constant	Value
pbTextUnitCell	12
pbTextUnitCharacter	1
pbTextUnitCharFormat	13
pbTextUnitCodePoint	17
pbTextUnitColumn	9
pbTextUnitLine	5
pbTextUnitObject	16
pbTextUnitParaFormat	14
pbTextUnitParagraph	4
pbTextUnitRow	10
pbTextUnitScreen	7
pbTextUnitSection	8
pbTextUnitSentence	3
pbTextUnitStory	6

pbTextUnitTable	15
pbTextUnitWindow	11
pbTextUnitWord	2

PbTrackingPresetType

Constant	Value
pbTrackingCustom	-1
pbTrackingLoose	1
pbTrackingMixed	-2
pbTrackingNormal	2
pbTrackingTight	3
pbTrackingVeryLoose	0
pbTrackingVeryTight	4

PbUnderlineType

Constant	Value
pbUnderlineDash	6
pbUnderlineDashHeavy	12
pbUnderlineDashLong	15
pbUnderlineDashLongHeavy	16
pbUnderlineDotDash	7
pbUnderlineDotDashHeavy	13
pbUnderlineDotDotDash	8
pbUnderlineDotDotDashHeavy	14
pbUnderlineDotHeavy	11
pbUnderlineDotted	4
pbUnderlineDouble	3
pbUnderlineMixed	-1
pbUnderlineNone	0
pbUnderlineSingle	1
pbUnderlineThick	5
pbUnderlineWavy	9

pbUnderlineWavyDouble	17
pbUnderlineWavyHeavy	10
pbUnderlineWordsOnly	2

PbUnitType

Constant	Value
pbUnitCM	1
pbUnitEmu	4
pbUnitFeet	6
pbUnitHa	9
pbUnitInch	0
pbUnitKyu	8
pbUnitMeter	7
pbUnitPica	2
pbUnitPixel	10
pbUnitPoint	3
pbUnitTwip	5

PbVerticalPictureLocking

Constant	Value
pbVerticalLockingBottom	2
pbVerticalLockingNone	0
pbVerticalLockingStretch	3
pbVerticalLockingTop	1

PbVerticalTextAlignmentType

Constant	Value
pbVerticalTextAlignmentBottom	2
pbVerticalTextAlignmentCenter	1
pbVerticalTextAlignmentTop	0

PbWebControlType

Constant	Value
pbWebControlCheckBox	100
pbWebControlCommandButton	101
pbWebControlHotSpot	110
pbWebControlHTMLFragment	107
pbWebControlListBox	102
pbWebControlMultiLineTextBox	103
pbWebControlOptionButton	104
pbWebControlSingleLineTextBox	105
pbWebControlWebComponent	106

PbWindowState

Constant	Value
pbWindowStateMaximize	0
pbWindowStateMinimize	1
pbWindowStateNormal	2

PbWizard

Constant	Value
pbWizardAdvertisements	12
pbWizardAirplanes	23
pbWizardBanners	21
pbWizardBrochures	8
pbWizardBusinessCards	3
pbWizardBusinessForms	20
pbWizardCalendars	13
pbWizardCatalogs	161
pbWizardCertificates	62
pbWizardEmailActivityEvent	302
pbWizardEmailFeaturedProduct	304

pbWizardEmailLetter	300
pbWizardEmailNewsletter	39
pbWizardEmailProductList	303
pbWizardEmailSpeakerEvent	301
pbWizardEnvelopes	7
pbWizardFlyers	16
pbWizardGiftCertificates	63
pbWizardGreetingCard	40
pbWizardInvitation	41
pbWizardJapaneseAdvertisements	165
pbWizardJapaneseAirplanes	164
pbWizardJapaneseBanners	121
pbWizardJapaneseBrochures	92
pbWizardJapaneseBusinessCards	91
pbWizardJapaneseBusinessForms	123
pbWizardJapaneseCalendars	82
pbWizardJapaneseCatalogs	177
pbWizardJapaneseCertificates	119
pbWizardJapaneseEnvelopes	93
pbWizardJapaneseFlyers	94
pbWizardJapaneseGiftCertificates	122
pbWizardJapaneseGreetingCards	80
pbWizardJapaneseInvitations	81
pbWizardJapaneseLabels	118
pbWizardJapaneseLetterheads	95
pbWizardJapaneseMenus	116
pbWizardJapaneseNewsletters	117
pbWizardJapaneseOrigami	163
pbWizardJapanesePostcards	78
pbWizardJapanesePrograms	115
pbWizardJapaneseSigns	149
pbWizardJapaneseWebSites	120
pbWizardLabels	19

pbWizardLetterheads	6
pbWizardMenus	59
pbWizardNewsletters	9
pbWizardNone	0
pbWizardOrigami	22
pbWizardPostcards	10
pbWizardPrograms	76
pbWizardQuickPublications	179
pbWizardResumes	18
pbWizardSigns	17
pbWizardWebSiteBlank	203
pbWizardWebSiteHomePage	5
pbWizardWebSiteProductSales	201
pbWizardWebSiteServices	202
pbWizardWebSiteThreePage	200
pbWizardWithComplimentsCards	73
pbWizardWordDocument	189

PbWizardGroup

Constant	Value
pbWizardGroupAccentBox	151
pbWizardGroupAccessoryBar	154
pbWizardGroupAdvertisements	68
pbWizardGroupAttentionGetter	61
pbWizardGroupBarbells	52
pbWizardGroupBorders	155
pbWizardGroupBoxes	50
pbWizardGroupCalendars	77
pbWizardGroupCheckerboards	53
pbWizardGroupCoupon	60
pbWizardGroupDots	49
pbWizardGroupEastAsiaZipCode	181

pbWizardGroupWebButtonsLink	136
pbWizardGroupWebCalendars	35
pbWizardGroupWebMastheads	102
pbWizardGroupWebNavigationBars	75
pbWizardGroupWebSidebars	104
pbWizardGroupWellPullQuotes	103

PbWizardNavBarAlignment

Constant	Value
pbnbAlignCenter	2
pbnbAlignLeft	1
pbnbAlignRight	3

PbWizardNavBarButtonStyle

Constant	Value
pbnbButtonStyleLarge	2
pbnbButtonStyleSmall	1
pbnbButtonStyleText	3

PbWizardNavBarDesign

Constant	Value
pbnbDesignAmbient	2
pbnbDesignBaseline	26
pbnbDesignBracket	11
pbnbDesignBulletStaff	20
pbnbDesignCapsule	3
pbnbDesignCornice	15
pbnbDesignCounter	13
pbnbDesignDimension	8
pbnbDesignDottedArrow	9
pbnbDesignEdge	17

pbnbDesignEnclosedArrow	12
pbnbDesignEndCap	14
pbnbDesignHollowArrow	10
pbnbDesignKeyPunch	22
pbnbDesignOffset	7
pbnbDesignOutline	5
pbnbDesignRadius	6
pbnbDesignRectangle	1
pbnbDesignRoundBullet	23
pbnbDesignSquareBullet	24
pbnbDesignStaff	16
pbnbDesignTopBar	21
pbnbDesignTopDrawer	4
pbnbDesignTopLine	18
pbnbDesignUnderscore	19
pbnbDesignWatermark	25

PbWizardPageType

Constant	Value
pbWizardPageTypeCatalogBlank	35
pbWizardPageTypeCatalogCalendar	22
pbWizardPageTypeCatalogEightItemsOneColumn	33
pbWizardPageTypeCatalogEightItemsTwoColumns	34
pbWizardPageTypeCatalogFeaturedItem	24
pbWizardPageTypeCatalogForm	36
pbWizardPageTypeCatalogFourItemsAlignedPictures	30
pbWizardPageTypeCatalogFourItemsOffsetPictures	31
pbWizardPageTypeCatalogFourItemsSquaredPictures	32
pbWizardPageTypeCatalogOneColumnText	18
pbWizardPageTypeCatalogOneColumnTextPicture	19
pbWizardPageTypeCatalogTableOfContents	23
pbWizardPageTypeCatalogThreeItemsAlignedPictures	27

pbWizardPageTypeCatalogThreeItemsOffsetPictures	28
pbWizardPageTypeCatalogThreeItemsStackedPictures	29
pbWizardPageTypeCatalogTwoColumnsText	20
pbWizardPageTypeCatalogTwoColumnsTextPicture	21
pbWizardPageTypeCatalogTwoItemsAlignedPictures	25
pbWizardPageTypeCatalogTwoItemsOffsetPictures	26
pbWizardPageTypeNewsletter3Stories	1
pbWizardPageTypeNewsletterCalendar	2
pbWizardPageTypeNewsletterOrderForm	15
pbWizardPageTypeNewsletterResponseForm	16
pbWizardPageTypeNewsletterSignupForm	17
pbWizardPageTypeNone	-1
pbWizardPageTypeWebAboutUs	501
pbWizardPageTypeWebArticle	512
pbWizardPageTypeWebBlank	524
pbWizardPageTypeWebCalendarPage	504
pbWizardPageTypeWebCalendarWithLinks	800
pbWizardPageTypeWebContactUs	505
pbWizardPageTypeWebEmployee	507
pbWizardPageTypeWebEmployeeList	506
pbWizardPageTypeWebEmployeesWithLinks	802
pbWizardPageTypeWebFAQ	508
pbWizardPageTypeWebHome	509
pbWizardPageTypeWebInformational	502
pbWizardPageTypeWebJobs	510
pbWizardPageTypeWebLegal	511
pbWizardPageTypeWebLinks	518
pbWizardPageTypeWebList	503
pbWizardPageTypeWebOrderForm	525
pbWizardPageTypeWebPhoto	513
pbWizardPageTypeWebPhotoGallery	514
pbWizardPageTypeWebPhotosWithLinks	805
pbWizardPageTypeWebProduct	515

pbWizardPageTypeWebProductList	516
pbWizardPageTypeWebProductsWithLinks	801
pbWizardPageTypeWebProjectList	517
pbWizardPageTypeWebProjectsWithLinks	804
pbWizardPageTypeWebResponseForm	526
pbWizardPageTypeWebSeminar	519
pbWizardPageTypeWebService	521
pbWizardPageTypeWebServiceList	520
pbWizardPageTypeWebServicesWithLinks	803
pbWizardPageTypeWebSignupForm	527
pbWizardPageTypeWebSpecial	522

PbWizardTag

Constant	Value
pbWizardTagAddress	10
pbWizardTagAddressGroup	117
pbWizardTagBriefDescriptionCaption	1361
pbWizardTagBriefDescriptionGraphic	1359
pbWizardTagBriefDescriptionSummary	1353
pbWizardTagBriefDescriptionSummaryPrimary	1365
pbWizardTagBriefDescriptionTitle	1364
pbWizardTagBusinessDescription	685
pbWizardTagCustomerMailingAddress	560
pbWizardTagDate	1835
pbWizardTagEAPostalCodeBox	2151
pbWizardTagEAPostalCodeGroup	2150
pbWizardTagEAPostalCodeLine	2152
pbWizardTagFloatingGraphicCaption	1362
pbWizardTagHourTimeDateInformation	684
pbWizardTagJobTitle	115
pbWizardTagLinkedStoryPrimary	1354
pbWizardTagLinkedStorySecondary	1355

pbWizardTagLinkedStoryTertiary	1356
pbWizardTagList	1837
pbWizardTagLocation	488
pbWizardTagLogoGroup	5
pbWizardTagMainFloatingGraphic	1357
pbWizardTagMainGraphic	1833
pbWizardTagMainTitle	1832
pbWizardTagMapPicture	489
pbWizardTagMasthead	1831
pbWizardTagNewsletterTitle	1344
pbWizardTagOrganizationName	7
pbWizardTagOrganizationNameGroup	118
pbWizardTagPageNumber	1346
pbWizardTagPersonalName	8
pbWizardTagPersonalNameGroup	116
pbWizardTagPhoneFaxEmail	113
pbWizardTagPhoneFaxEmailGroup	120
pbWizardTagPhoneNumber	114
pbWizardTagPhotePlaceholderText	1135
pbWizardTagPhotoPlaceholderFrame	1134
pbWizardTagPublicationDate	1341
pbWizardTagQuickPubContent	2143
pbWizardTagQuickPubHeading	2140
pbWizardTagQuickPubMessage	2141
pbWizardTagQuickPubPicture	2142
pbWizardTagReturnAddressLines	793
pbWizardTagStampBox	887
pbWizardTagStampBoxOutline	794
pbWizardTagStory	1349
pbWizardTagStoryCaptionPrimary	1351
pbWizardTagStoryCaptionSecondary	1373
pbWizardTagStoryGraphicPrimary	1350
pbWizardTagStoryGraphicSecondary	1360

pbWizardTagStoryTitle	1348
pbWizardTagTableOfContents	1343
pbWizardTagTableOfContentsTitle	1342
pbWizardTagTagLine	112
pbWizardTagTagLineGroup	119
pbWizardTagTime	1836

PbWrapSideType

Constant	Value
pbWrapSideBoth	0
pbWrapSideLarger	3
pbWrapSideLeft	1
pbWrapSideMixed	-1
pbWrapSideNeither	4
pbWrapSideRight	2

PbWrapType

Constant	Value
pbWrapTypeMixed	-1
pbWrapTypeNone	0
pbWrapTypeSquare	1
pbWrapTypeThrough	3
pbWrapTypeTight	2
pbWrapTypeTopAndBottom	4

PbZoom

Constant	Value
pbZoomFitSelection	-3
pbZoomPageWidth	-1
pbZoomWholePage	-2

PrintPlate Property

Returns or sets **True** if the printable plate is set to print. The default is **True**. Read/write **Boolean**.

expression.**PrintPlate**

expression Required. An expression that returns a **PrintablePlate** object.

Remarks

This property corresponds to the **Print plate** check boxes on the **Separations** tab of the **Advanced Print Settings** dialog box.

Example

The following example returns a spot color plate and sets several of its properties. The example assumes that separations have been specified as the active publication's print mode.

```
Sub SetPlatePropertiesByInkName()

Dim pplPlate As PrintablePlate
ActiveDocument.AdvancedPrintOptions.UseCustomHalftone = True

    Set pplPlate = ActiveDocument.AdvancedPrintOptions.PrintablePlat

    With pplPlate
        .Angle = 75
        .Frequency = 133
        .PrintPlate = True
    End With

End Sub
```

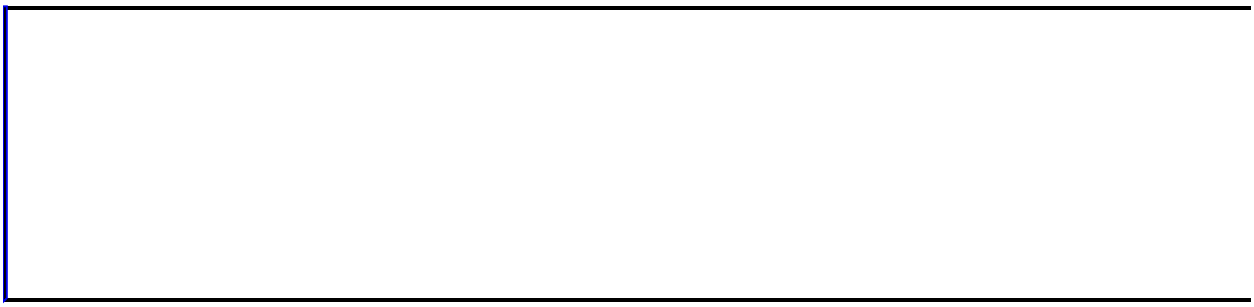
Returning an Object from a Collection

The **Item** method returns a single object from a collection. The following example sets a variable to a **Page** object that represents the first page in the **Pages** collection.

```
Sub SetFirstPage()
    Dim pgFirst As Page
    Set pgFirst = ActiveDocument.Pages.Item(1)
End Sub
```

The **Item** method is the default method for most collections, so you can write the same statement more concisely by omitting the **Item** keyword.

```
Sub SetFirstPage()
    Dim pgFirst As Page
    Set pgFirst = ActiveDocument.Pages(1)
End Sub
```



Using Events with the Document Object

The **Document** object supports seven events: BeforeClose, Open, Redo, ShapesAdded, ShapesRemoved, Undo, and WizardAfterChange. You write procedures to respond to these events in the class module named "ThisDocument." Use the following steps to create an event procedure.

1. Under your publication project in the Project Explorer window, double-click **ThisDocument**. (In Folder view, **ThisDocument** is located in the **Microsoft Publisher Objects** folder.)
2. Select **Document** from the **Object** drop-down list box.
3. Select an event from the **Procedure** drop-down list box.

   An empty subroutine is added to the class module.

4. Add the Visual Basic instructions you want to run when the event occurs.

Example

This example shows an Open event procedure that displays a message when a publication is opened.

```
Private Sub Document_Open()
    MsgBox "This publication is copyrighted."
End Sub
```

The following example shows a BeforeClose event procedure that prompts the user for a yes or no response before closing a document.

```
Private Sub Document_BeforeClose(Cancel As Boolean)
    Dim intResponse As Integer

    intResponse = MsgBox("Do you really want to close " _
        & "the document?", vbYesNo)

    If intResponse = vbNo Then Cancel = True
End Sub
```

Note For information on creating event procedures for the **Application** object, see Using Events with the Application Object.

Using Events with the Application Object

To create an event handler for an event of the **Application** object, you need to complete the following three steps:

1. Declare an object variable in a class module to respond to the events.
2. Write the specific event procedures.
3. Initialize the declared object from another module.

Declare the Object Variable

Before you can write procedures for the events of the **Application** object, you must create a new class module and declare an object of type **Application** with events. For example, assume that a new class module is created and called EventClassModule. The new class module contains the following code.

```
Public WithEvents App As Publisher.Application
```

Write the Event Procedures

After the new object has been declared with events, it appears in the **Object** drop-down list box in the class module, and you can write event procedures for the new object. (When you select the new object in the **Object** box, the valid events for that object are listed in the **Procedure** drop-down list box.) Select an event from the **Procedure** drop-down list box; an empty procedure is added to the class module.

```
Private Sub App_DocumentOpen()

End Sub
```

Initialize the Declared Object

Before the procedure will run, you must connect the declared object in the class module (App in this example) with the **Application** object. You can do this with the following code from any module.

```
Dim X As New EventClassModule
Sub Register_Event_Handler()
    Set X.App = Publisher.Application
End Sub
```

Run the Register_Event_Handler procedure. After the running procedure, the App object in the class module points to the Microsoft Publisher **Application** object, and the event procedures in the class module will run when the events occur.

Note For information on creating event procedures for the **Document** object, see Using Events with the Document Object.