

Microsoft Word Objects



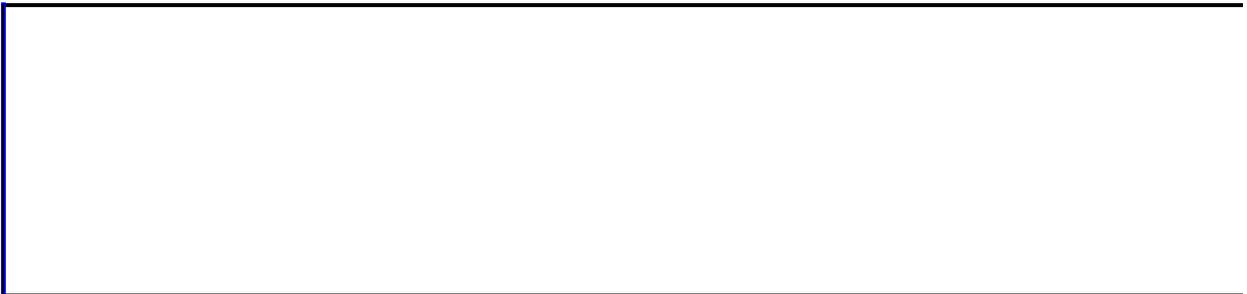
- └─ [Style](#)
- └─ [FileConverters](#)
 - └─ [FileConverter](#)
- └─ [FileDialog](#)
- └─ [FileSearch](#)
- └─ [FontNames](#)
- └─ [HangulHanjaConversionDictionaries](#)
 - └─ [Dictionary](#)
- └─ [KeyBindings](#)
 - └─ [KeyBinding](#)
- └─ [KeysBoundTo](#)
 - └─ [KeyBinding](#)
- └─ [Languages](#)
 - └─ [Language](#)
 - └─ [Dictionary](#)
- └─ [LanguageSettings](#)
- └─ [ListGalleries](#)
 - └─ [ListGallery](#)
 - └─ [ListTemplates](#) ▶
- └─ [MailingLabel](#)
 - └─ [CustomLabels](#)
 - └─ [CustomLabel](#)
- └─ [MailMessage](#)
- └─ [NewFile](#)
- └─ [Options](#)
- └─ [RecentFiles](#)
 - └─ [RecentFile](#)
- └─ [Selection](#) ▶
- └─ [SpellingSuggestions](#)
 - └─ [SpellingSuggestion](#)

- | [SynonymInfo](#)
- | [System](#)
- | [TaskPanels](#)
 - | [TaskPane](#)
- | [Tasks](#)
 - | [Task](#)
- | [Templates](#) ▶
- | [Windows](#) ▶

Legend

Object and collection
Object only

▶ Click red arrow to expand chart



What's New for Microsoft Word 2002 Developers

Extensive changes have been made to the Microsoft Word 2002 Visual Basic object model to support new and improved features in the application.

Visit the [Office Developer Center](#) at MSDN Online for the latest Microsoft Word development information, including new technical articles, downloads, samples, product news, and more.

New Language Elements

The following topics provide lists of language elements that are new in Word 2002.

[New Objects](#)

[New Properties \(by Object\)](#)

[New Properties \(Alphabetic List\)](#)

[New Methods \(by Object\)](#)

[New Methods \(Alphabetic List\)](#)

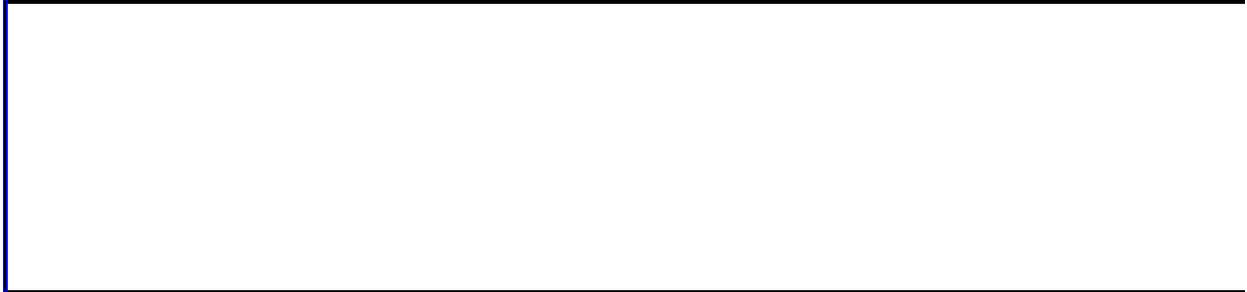
[New Events](#)

[Language-Specific Properties and Methods](#)

Hidden Language Elements

The following topic provides a list of properties that have been hidden in Word 2002.

[Hidden Properties](#)



Understanding Objects, Properties, and Methods

Objects are the fundamental building block of Visual Basic; nearly everything you do in Visual Basic involves modifying objects. Every element of Microsoft Word — documents, tables, paragraphs, bookmarks, fields and so on — can be represented by an object in Visual Basic.

What are objects and collections?

An object represents an element of Word, such as a document, a paragraph, a bookmark, or a single character. A collection is an object that contains several other objects, usually of the same type; for example, all the bookmark objects in a document are contained in a single collection object. Using properties and methods, you can modify a single object or an entire collection of objects.

What is a property?

A property is an attribute of an object or an aspect of its behavior. For example, properties of a document include its name, its content, and its save status, as well as whether change tracking is turned on. To change the characteristics of an object, you change the values of its properties.

To set the value of a property, follow the reference to an object with a period, the property name, an equal sign, and the new property value. The following example turns on change tracking in the document named "MyDoc.doc."

```
Sub TrackChanges()  
    Documents("Sales.doc").TrackRevisions = True  
End Sub
```

In this example, `Documents` refers to the collection of open documents, and the name "Sales.doc" identifies a single document in the collection. The [TrackRevisions](#) property is set for that single document.

Some properties cannot be set. The Help topic for a property indicates whether that property can be set (read-write) or can only be read (read-only).

You can return information about an object by returning the value of one of its properties. The following example returns the name of the active document.

```
Sub GetDocumentName()  
    Dim strDocName As String  
    strDocName = ActiveDocument.Name  
    MsgBox strDocName  
End Sub
```

In this example, `ActiveDocument` refers to the document in the active window in Word. The name of that document is assigned to the variable `strDocName`.

Remarks

The Help topic for each property indicates whether you can set that property (read-write), only read the property (read-only), or only write the property (write-only). Also the Object Browser in the Visual Basic Editor displays the read-write status at the bottom of the browser window when the property is

selected.

What is a method?

A method is an action that an object can perform. For example, just as a document can be printed, the [Document](#) object has a [PrintOut](#) method. Methods often have arguments that qualify how the action is performed. The following example prints the first three pages of the active document.

```
Sub PrintThreePages()  
    ActiveDocument.PrintOut Range:=wdPrintRangeOfPages, Pages:="1-3"  
End Sub
```

In most cases, methods are actions and properties are qualities. Using a method causes something to happen to an object, while using a property returns information about the object or it causes a quality about the object to change.

Returning an object

Most objects are returned by returning a single object from the collection. For example, the [Documents](#) collection contains the open Word documents. You use the [Documents](#) property of the [Application](#) object (the object at the top of the Word object hierarchy) to return the **Documents** collection.

After you've accessed the collection, you can return a single object by using an index value in parentheses (this is similar to how you work with arrays). The index value is usually a number or a name. For more information, see [Returning an Object from a Collection](#).

The following example uses the **Documents** property to access the **Documents** collection. The index number is used to return the first document in the **Documents** collection. The [Close](#) method is then applied to the **Document** object to close the first document in the **Documents** collection.

```
Sub CloseDocument()  
    Documents(1).Close  
End Sub
```

The following example uses a name (specified as a string) to identify a **Document** object within the **Documents** collection.

```
Sub CloseSalesDoc()  
    Documents("Sales.doc").Close  
End Sub
```

Collection objects often have methods and properties which you can use to modify the entire collection of objects. The **Documents** object has a [Save](#) method that saves all the documents in the collection. The following example saves the open documents by applying the Save method.

```
Sub SaveAllOpenDocuments()  
    Documents.Save  
End Sub
```

The **Document** object also has a **Save** method available for saving a single document. The following example saves the document named Sales.doc.

```
Sub SaveSalesDoc()  
    Documents("Sales.doc").Save  
End Sub
```

To return an object that is further down in the Word object hierarchy, you must "drill down" to it by using properties and methods to return objects.

To see how this is done, open the Visual Basic Editor and click **Object Browser** on the **View** menu. Click **Application** in the **Classes** list on the left. Then click **ActiveDocument** from the list of members on the right. The text at bottom of the Object Browser indicates that **ActiveDocument** is a read-only property that returns a **Document** object. Click **Document** at the bottom of the Object Browser; the **Document** object is automatically selected in the **Classes** list, and the **Members** list displays the members of the **Document** object. Scroll through the list of members until you find **Close**. Click the **Close** method. The text at the bottom of the Object Browser window shows the syntax for the method. For more information about the method, press F1 or click the **Help** button to jump to the **Close** method Help topic.

Given this information, you can write the following instruction to close the active document.

```
Sub CloseDocSaveChanges()  
    ActiveDocument.Close SaveChanges:=wdSaveChanges  
End Sub
```

The following example maximizes the active document window.

```
Sub MaximizeDocumentWindow()  
    ActiveDocument.ActiveWindow.WindowState = wdWindowStateMaximize  
End Sub
```

The **ActiveWindow** property returns a **Window** object that represents the active window. The **WindowState** property is set to the maximize constant (**wdWindowStateMaximize**).

The following example creates a new document and displays the Save As dialog box so that a name can be provided for the document.

```
Sub CreateSaveNewDocument()  
    Documents.Add.Save  
End Sub
```

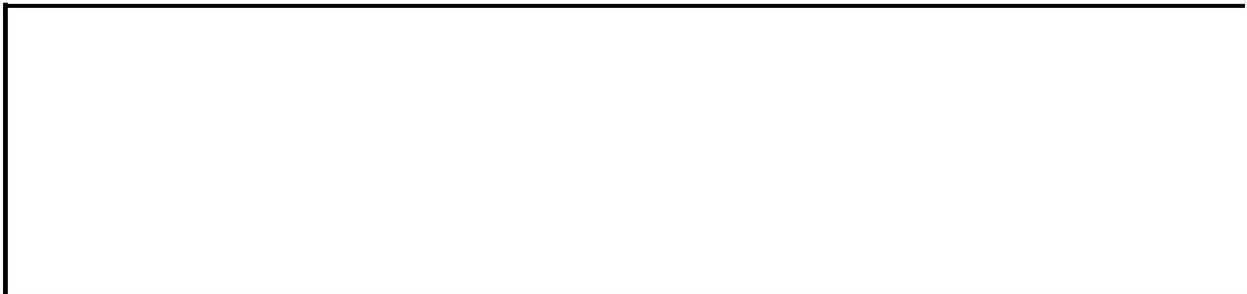
The **Documents** property returns the **Documents** collection. The [Add](#) method creates a new document and returns a **Document** object. The **Save** method is then applied to the **Document** object.

As you can see, you use methods or properties to drill down to an object. That is, you return an object by applying a method or property to an object above it in the object hierarchy. After you return the object you want, you can apply the methods and control the properties of that object. To review the hierarchy of objects, see [Microsoft Word Objects](#).

Getting Help on objects, methods, and properties

Until you become familiar with the Word object model, there are a few tools you can use to help you to drill down through the hierarchy.

- **Auto List Members.** When you type a period (.) after an object in the Visual Basic Editor, a list of available properties and methods is displayed. For example, if you type **Application.**, a drop-down list of methods and properties of the **Application** object is displayed.
- **Help.** You can also use Help to find out which properties and methods can be used with an object. Each object topic in Help includes a See Also jump that displays a list of properties and methods for the object. Press F1 in the Object Browser or a module to jump to the appropriate Help topic.
- **[Microsoft Word Objects](#).** This topic illustrates how Word objects are arranged in the hierarchy. Click an object in the graphic to display the corresponding Help topic.
- **Object Browser.** The Object Browser in the Visual Basic Editor displays the members (properties and methods) of the Word objects.



Frequently Asked Visual Basic Questions

General questions

[How do I convert my WordBasic macros to Visual Basic?](#)

[How do I find out the Visual Basic equivalent for a WordBasic command or function?](#)

[How do I record macros?](#)

[What are objects, properties and methods?](#)

[How do I find out which property or method I need?](#)

[How do I return a single object from a collection?](#)

Questions about Word features

[How do I refer to the active element \(for example, paragraph, table, field\)?](#)

[What is a Range object?](#)

[How do I refer to words, sentences, paragraphs, or sections in a document?](#)

[I keep getting the "object doesn't support this property or method" error; how can I avoid it?](#)

[How do I create, open, save and close documents?](#)

[How do I select text in a document?](#)

[How do I insert text into a document?](#)

[I keep getting the "requested member of the collection does not exist" error; how can I avoid it?](#)

[How do I loop on a collection?](#)

[How do I prompt for information from the user?](#)

[How do I return text from a document?](#)

[How do I know if the **Application** property is needed before a top level property or method?](#)

[How do I display a built-in Microsoft Word dialog box?](#)

[I keep getting an error when I try to access a table row or column?](#)



Automating Common Word Tasks

This topic includes some common Microsoft Word tasks and the Visual Basic code needed to accomplish the tasks.

[Applying formatting to text](#)

[Editing text](#)

[Finding and replacing text or formatting](#)

[Miscellaneous tasks](#)

[Working with tables](#)

[Working with documents](#)



Referring to the Active Document Element

To refer to the active paragraph, table, field, or other document element, use the [Selection](#) property to return a [Selection](#) object. From the **Selection** object, you can access all paragraphs in the selection or the first paragraph in the selection. The following example applies a border around the first paragraph in the selection.

```
Sub BorderAroundFirstParagraph()  
    Selection.Paragraphs(1).Borders.Enable = True  
End Sub
```

The following example applies a border around each paragraph in the selection.

```
Sub BorderAroundSelection()  
    Selection.Paragraphs.Borders.Enable = True  
End Sub
```

The following example applies shading to the first row of the first table in the selection.

```
Sub ShadeTableRow()  
    Selection.Tables(1).Rows(1).Shading.Texture = wdTexture10Percent  
End Sub
```

An error occurs if the selection doesn't include a table. Use the [Count](#) property to determine if the selection includes a table. The following example applies shading to the first row of the first table in the selection.

```
Sub ShadeTableRow()  
    If Selection.Tables.Count >= 1 Then  
        Selection.Tables(1).Rows(1).Shading.Texture = wdTexture25Per  
    Else  
        MsgBox "Selection doesn't include a table"  
    End If  
End Sub
```

The following example applies shading to the first row of every table in the selection. The **For Each...Next** loop is used to step through the individual tables in the selection.

```
Sub ShadeAllFirstRowsInTables()  
    Dim tblTable As Table  
    If Selection.Tables.Count >= 1 Then  
        For Each tblTable In Selection.Tables  
            tblTable.Rows(1).Shading.Texture = wdTexture30Percent  
        Next tblTable  
    End If  
End Sub
```



Storing Values When a Macro Ends

When a macro ends, the values stored in its variables aren't automatically saved to disk. If a macro needs to preserve a value, it must store that value outside itself before the macro execution is completed. This topic describes five locations where macro values can be easily stored and retrieved.

Document variables

Document variables allow you to store values as part of a document or a template. For example, you might store macro values in the document or template where the macro resides. You can add variables to a document or template using the [Add](#) method of the [Variables](#) collection. The following example saves a document variable in the same location as the macro that is running (document or template) using the **ThisDocument** property.

```
Sub AddDocumentVariable()  
    ThisDocument.Variables.Add Name:="Age", Value:=12  
End Sub
```

The following example uses the [Value](#) property with a [Variable](#) object to return the value of a document variable.

```
Sub UseDocumentVariable()  
    Dim intAge As Integer  
    intAge = ThisDocument.Variables("Age").Value  
End Sub
```

Remarks

You can use the DOCVARIABLE field to insert a document variable into a document.

Document properties

Like document variables, document properties allow you to store values as part of a document or a template. Document properties can be viewed in the **Properties** dialog box (**File** menu).

The Word object model breaks document properties into two groups: built-in and custom. Custom document properties include the properties shown on the **Custom** tab in the **Properties** dialog box. Built-in document properties include the properties on all the tabs in the **Properties** dialog box except the **Custom** tab.

To access built-in properties, use the [BuiltInDocumentProperties](#) property to return a [DocumentProperties](#) collection that includes the built-in document properties. Use the [CustomDocumentProperties](#) property to return a **DocumentProperties** collection that includes the custom document properties. The following example creates a custom document property named "YourName" in the same location as the macro that is running (document or template).

```
Sub AddCustomDocumentProperties()  
    ThisDocument.CustomDocumentProperties.Add Name:="YourName", _  
        LinkToContent:=False, Value:="Joe", Type:=msoPropertyTypeStr  
End Sub
```

Built-in document properties cannot be added to the **DocumentProperties** collection returned by the **BuiltInDocumentProperties** property. You can, however, retrieve the contents of a built-in document property or change the value of a read/write built-in document property.

Remarks

You can use the DOCPROPERTY field to insert document properties into a document.

AutoText entries

AutoText entries can be used to store information in a template. Unlike a document variable or property, AutoText entries can include items beyond macro variables such as formatted text or a graphic. Use the [Add](#) method with the [AutoTextEntries](#) collection to create a new AutoText entry. The following example creates an AutoText entry named "MyText" that contains the contents of the selection. If the following instruction is part of a template macro, the new AutoText entry is stored in the template, otherwise, the AutoText entry is stored in the template attached to the document where the instruction resides.

```
Sub AddAutoTextEntry()  
    ThisDocument.AttachedTemplate.AutoTextEntries.Add Name:="MyText"  
        Range:=Selection.Range  
End Sub
```

Use the [Value](#) property with an [AutoTextEntry](#) object to retrieve the contents of an AutoText entry object.

Settings files

You can set and retrieve information from a settings file using the [PrivateProfileString](#) property. The structure of a Windows settings file is the same as the Windows 3.1 WIN.INI file. The following example sets the DocNum key to 1 under the DocTracker section in the Macro.ini file.

```
Sub MacroSystemFile()  
    System.PrivateProfileString( _  
        FileName:="C:\My Documents\Macro.ini", _  
        Section:="DocTracker", Key:="DocNum") = 1  
End Sub
```

After running the above instruction, the Macro.ini file includes the following text.

```
[DocTracker]  
DocNum=1
```

The **PrivateProfileString** property has three arguments: *FileName*, *Section*, and *Key*. The *FileName* argument is used to specify a settings file path and file name. The *Section* argument specifies the section name that appears between brackets before the associated keys (don't include the brackets with section name). The *Key* argument specifies the key name which is followed by an equal sign (=) and the setting.

Use the same **PrivateProfileString** property to retrieve a setting from a settings file. The following example retrieves the DocNum setting under the DocTracker section in the Macro.ini file.

```
Sub GetSystemFileInfo()  
    Dim intDocNum As Integer  
    intDocNum = System.PrivateProfileString( _  
        FileName:="C:\My Documents\Macro.ini", _  
        Section:="DocTracker", Key:="DocNum")  
    MsgBox "DocNum is " & intDocNum  
End Sub
```

Windows registry

You can set and retrieve information from the Windows registry using the [PrivateProfileString](#) property. The following example retrieves the Microsoft Word 2002 program directory from the Windows registry.

```
Sub GetRegistryInfo()  
    Dim strSection As String  
    Dim strPgmDir As String  
    strSection = "HKEY_CURRENT_USER\Software\Microsoft" _  
        & "\Office\10.0\Word\Options"  
    strPgmDir = System.PrivateProfileString(FileName:= "", _  
        Section:=strSection, Key:="PROGRAMDIR")  
    MsgBox "The directory for Word is - " & strPgmDir  
End Sub
```

The **PrivateProfileString** property has three arguments: **FileName**, **Section**, and **Key**. To return or set a value for a registry entry, specify an empty string ("") for the **FileName** argument. The **Section** argument should be the complete path to the registry subkey. The **Key** argument should be the name of an entry in the subkey specified by **Section**.

You can also set information in the Windows registry using the following **PrivateProfileString** syntax.

System.PrivateProfileString(FileName, Section, Key) = value

The following example sets the DOC-PATH entry to “C:\My Documents” in the Options subkey for Word 2002 in the Windows registry.

```
Sub SetDocumentDirectory()  
    Dim strDocDirectory As String  
    strDocDirectory = "HKEY_CURRENT_USER\Software\Microsoft" _  
        & "\Office\10.0\Word\Options"  
    System.PrivateProfileString(FileName:= "", _  
        Section:=strDocDirectory, Key:="DOC-PATH") = "C:\My Document  
End Sub
```



Built-in Dialog Box Argument Lists

Many of the built-in dialog boxes in Microsoft Word have options that you may want to set. To set or return the properties associated with a Word dialog box, use the equivalent Visual Basic properties and methods. For example, if you want to print a document, use the Word Visual Basic for Applications [PrintOut](#) method. The following code prints the current document using the **Print** dialog box default settings. However, if you don't want to use the default setting in the print dialog, you can use the arguments associated with the **PrintOut** method.

```
Sub PrintCurrentDocument()  
    ThisDocument.PrintOut  
End Sub
```

Although you are encouraged to use VBA keywords to get or set the value of dialog box options, many of the built-in Word dialog boxes have arguments that you can also use to set or get values from a dialog box. For more information, see [Displaying built-in Word dialog boxes](#).

WdWordDialog constant	Argument list(s)
wdDialogConnect	<i>Drive, Path, Password</i>
wdDialogConsistencyChecker	(none)
wdDialogControlRun	<i>Application</i>
wdDialogConvertObject	<i>IconNumber, ActivateAs, IconFileName, Caption, Class, DisplayIcon, Floating</i>
wdDialogCopyFile	<i>FileName, Directory</i>
wdDialogCreateAutoText	(none)
wdDialogCSSLinks	(none)
wdDialogDocumentStatistics	<i>FileName, Directory, Template Created, LastSaved, LastSave, Revision, Time, Printed, Page Characters, Paragraphs, Line Horizontal, Vertical, Relative</i>
wdDialogDrawAlign	

wdDialogDrawSnapToGrid

*SnapToGrid, XGrid, YGrid, X
YOrigin, SnapToShapes, XGr
YGridDisplay, FollowMargin
ViewGridLines,
DefineLineBasedOnGrid*

wdDialogEditAutoText

*Name, Context, InsertAs, Inse
Define, InsertAsText, Delete,
CompleteAT*

wdDialogEditCreatePublisher

(For information about this con
consult the language reference
included with Microsoft Office
Macintosh Edition.)

wdDialogEditFind

*Find, Replace, Direction, Mat
WholeWord, PatternMatch,
SoundsLike, FindNext, Repla
ReplaceAll, Format, Wrap,
FindAllWordForms, MatchBy
FuzzyFind, Destination, Corr
MatchKashida, MatchDiacriti
MatchAlefHamza, MatchCon*

wdDialogEditFrame

*WidthRule, LockAnchor, Heig
Find, Replace, Direction, Mat
WholeWord, PatternMatch,
SoundsLike, FindNext, Repla*

wdDialogEditGoTo

*ReplaceAll, Format, Wrap,
FindAllWordForms, MatchBy
FuzzyFind, Destination, Corr
MatchKashida, MatchDiacriti
MatchAlefHamza, MatchCon*

wdDialogEditGoToOld

(none)

wdDialogEditLinks

*UpdateMode, Locked,
SavePictureInDoc, UpdateNo
OpenSource, KillLink, Link,
Application, Item, FileName*

wdDialogEditObject

Verb

wdDialogEditPasteSpecial

*IconNumber, Link, DisplayIc
DataType, IconFileName, Ca*

wdDialogEditPublishOptions

Floating

(For information about this control consult the language reference included with Microsoft Office Macintosh Edition.)

wdDialogEditReplace

Find, Replace, Direction, MatchWholeWord, PatternMatch, SoundsLike, FindNext, ReplaceAll, Format, Wrap, FindAllWordForms, MatchByFuzzyFind, Destination, CorrectMatchKashida, MatchDiacriticalMatchAlefHamza, MatchCon

wdDialogEditStyle

(none)

wdDialogEditSubscribeOptions

(For information about this control consult the language reference included with Microsoft Office Macintosh Edition.)

wdDialogEditSubscribeTo

(For information about this control consult the language reference included with Microsoft Office Macintosh Edition.)

wdDialogEditTOACategory

Category, CategoryName

wdDialogEmailOptions

(none)

wdDialogFileDocumentLayout

Tab, PaperSize, TopMargin, BottomMargin, LeftMargin, RightMargin, Gutter, PageWidth, PageHeight, Orientation, FirstPage, OtherPages, VertAlign, ApplyDefault, FacingPages, HeaderDistance, FooterDistance, SectionStart, OddAndEvenPages, DifferentFirstPage, Endnotes, LineNum, StartingNum, FrontPage, CountBy, NumMode, TwoOnPage, GutterPosition, LayoutMode, CharsLine, LinesPage, CharF

wdDialogFileFind

LinePitch, DocFontName, DocFontSize, PageColumns, FirstPageOnLeft, SectionType, RTLAlignment

SearchName, SearchPath, NaSubDir, Title, Author, Keyword, Subject, Options, MatchCase, PatternMatch, DateSavedFrom, DateSavedTo, SavedBy, DateCreatedFrom, DateCreatedTo, View, SortBy, ListBy, Selected, Delete, ShowFolders, MatchB

(For information about this control, consult the language reference included with Microsoft Office Macintosh Edition.)

wdDialogFileMacPageSetup

wdDialogFileNew

Template, NewTemplate, DocumentType, Visible

wdDialogFileOpen

Name, ConfirmConversions, 1, LinkToSource, AddToMru, PasswordDoc, PasswordDot, 1, WritePasswordDoc, WritePassword, Connection, SQLStatement, SQLStatement1, Format, EncVisible

Tab, PaperSize, TopMargin, BottomMargin, LeftMargin, RightMargin, Gutter, PageWidth, PageHeight, Orientation, FirstPage, OtherPages, VertAlign, Apply, Default, FacingPages, HeaderDistance, SectionStart, OddAndEvenPages,

wdDialogFilePageSetup

DifferentFirstPage, Endnotes, LineNum, StartingNum, FromPage, CountBy, NumMode, TwoOnOne, GutterPosition, LayoutMode, CharsLine, LinesPage, CharF

wdDialogFilePrint

LinePitch, DocFontName, DocFontSize, PageColumns, FirstPageOnLeft, SectionType, RTLAlignment

Background, AppendPrFile, PrToFileName, From, To, Type, NumCopies, Pages, Order, PrintCollate, FileName, Printer, OutputPrinter, DuplexPrint, PrintZoomColumn, PrintZoomPrintZoomPaperWidth, PrintZoomPaperHeight

(For information about this control, consult the language reference included with Microsoft Office Macintosh Edition.)

wdDialogFilePrintOneCopy

wdDialogFilePrintSetup

Printer, Options, Network, DoNotSetAsSysDefault

Subject, Message, AllAtOnce, ReturnWhenDone, TrackStatus, Protect, AddSlip, RouteDocument, AddRecipient, OldRecipient, ClearSlip, ClearRecipients, AddName, Format, LockAnnot, PrintAddToMru, WritePassword, RecommendReadOnly, EmbedNativePictureFormat, FormsSaveAsAOCELetter, WriteVersionDesc

wdDialogFileRoutingSlip

wdDialogFileSaveAs

Name, Format, LockAnnot, PrintAddToMru, WritePassword, RecommendReadOnly, EmbedNativePictureFormat, FormsSaveAsAOCELetter, WriteVersionDesc

wdDialogFileSaveVersion

(none)

wdDialogFileSummaryInfo

Title, Subject, Author, Keywords, Comments, FileName, DirectTemplate, CreateDate, LastSavedBy, RevisionNumber, EditTime, LastPrintedDate, NumWords, NumChars, NumLines, Update, FileSize

wdDialogFileVersions

wdDialogFitText

wdDialogFontSubstitution

wdDialogFormatAddrFonts

wdDialogFormatBordersAndShading

AutoVersion, VersionDesc

FitTextWidth

UnavailableFont, SubstituteF

Points, Underline, Color,

StrikeThrough, Superscript, S

Hidden, SmallCaps, AllCaps,

Position, Kerning, KerningMi

Default, Tab, Font, Bold, Itali

DoubleStrikeThrough, Shado

Outline, Emboss, Engrave, Sc

Animations, CharAccent, For

FontLowAnsi, FontHighAnsi,

CharacterWidthGrid, ColorR

UnderlineColor, PointsBi, Co

FontNameBi, BoldBi, ItalicBi

DiacColor

ApplyTo, Shadow, TopBorder,

LeftBorder, BottomBorder,

RightBorder, HorizBorder, Ve

TopColor, LeftColor, BottomC

RightColor, HorizColor, Vert

FromText, Shading, Foregrou

Background, Tab, FineShadir

TopStyle, LeftStyle, BottomSty

RightStyle, HorizStyle, VertSty

TopWeight, LeftWeight, Botto

RightWeight, HorizWeight, Ve

BorderObjectType, BorderArt

BorderArt, FromTextTop,

FromTextBottom, FromTextL

FromTextRight, OffsetFrom, .

SurroundHeader, SurroundF

JoinBorder, LineColor, Whicl

TL2BRBorder, TR2BLBorder

TL2BRColor, TR2BLColor,

TL2BRStyle, TR2BLStyle,

TL2BRWeight, TR2BLWeight

ForegroundRGB, Backgroun

	<i>TopColorRGB, LeftColorRGB, BottomColorRGB, RightColorRGB, HorizColorRGB, VertColorRGB, TL2BRColorRGB, TR2BLColorRGB, LineColorRGB</i>
wdDialogFormatBulletsAndNumbering	(none)
wdDialogFormatCallout	<i>Type, Gap, Angle, Drop, Length, Border, AutoAttach, Accent</i>
wdDialogFormatChangeCase	<i>Type</i>
wdDialogFormatColumns	<i>Columns, ColumnNo, ColumnSpacing, EvenlySpace, ApplyColsTo, ColLine, StartN, FlowColumnsRtl</i>
	<i>ApplyTo, Shadow, TopBorder, LeftBorder, BottomBorder, RightBorder, HorizBorder, VertBorder, TopColor, LeftColor, BottomColor, RightColor, HorizColor, VertColor, FromText, Shading, ForegroundBackground, Tab, FineShading, TopStyle, LeftStyle, BottomStyle, RightStyle, HorizStyle, VertStyle, TopWeight, LeftWeight, BottomWeight, RightWeight, HorizWeight, VertWeight, BorderObjectType, BorderArt, BorderArtType, FromTextTop, FromTextBottom, FromTextLeft, FromTextRight, OffsetFrom, SurroundHeader, SurroundFootnote, JoinBorder, LineColor, WhichColor, TL2BRBorder, TR2BLBorder, TL2BRColor, TR2BLColor, TL2BRStyle, TR2BLStyle, TL2BRWeight, TR2BLWeight, ForegroundRGB, BackgroundRGB, TopColorRGB, LeftColorRGB, BottomColorRGB, RightColorRGB, HorizColorRGB, VertColorRGB</i>
wdDialogFormatDefineStyleBorders	

wdDialogFormatDefineStyleFont

*TL2BRColorRGB, TR2BLCo
LineColorRGB*

*Points, Underline, Color,
StrikeThrough, Superscript, S
Hidden, SmallCaps, AllCaps,
Position, Kerning, KerningMi
Default, Tab, Font, Bold, Itali
DoubleStrikeThrough, Shado
Outline, Emboss, Engrave, Sc
Animations, CharAccent, For
FontLowAnsi, FontHighAnsi,
CharacterWidthGrid, ColorR
UnderlineColor, PointsBi, Co
FontNameBi, BoldBi, ItalicBi
DiacColor*

wdDialogFormatDefineStyleFrame

*Wrap, WidthRule, FixedWidth
HeightRule, FixedHeight,
PositionHorz, PositionHorzRe
DistFromText, PositionVert,
PositionVertRel, DistVertFron
MoveWithText, LockAnchor,
RemoveFrame*

wdDialogFormatDefineStyleLang

*Language, CheckLanguage, 1
NoProof*

wdDialogFormatDefineStylePara

*LeftIndent, RightIndent, Befo
LineSpacingRule, LineSpacin
Alignment, WidowControl,
KeepWithNext, KeepTogether,
PageBreak, NoLineNum, Dor
Tab, FirstIndent, OutlineLeve
Kinsoku, WordWrap, Overflow
TopLinePunct, AutoSpaceDE
LineHeightGrid, AutoSpaceD
CharAlign, CharacterUnitLef
AdjustRight, CharacterUnitFi
CharacterUnitRightIndent,
LineUnitBefore, LineUnitAfte
OrientationBi*

wdDialogFormatDefineStyleTabs

*Position, DefTabs, Align, Lea
Clear, ClearAll*

wdDialogFormatDrawingObject

*Left, PositionHorzRel, Top,
PositionVertRel, LockAnchor,
FloatOverText, WrapSide,
TopDistanceFromText,
BottomDistanceFromText,
LeftDistanceFromText,
RightDistanceFromText, Wra
HRWidthType, HRHeight, HI
HRAAlign, Text, AllowOverlap,
HorizRule*

wdDialogFormatDropCap

*Position, Font, DropHeight,
DistFromText*

wdDialogFormatEncloseCharacters

*Style, Text, Enclosure
Points, Underline, Color,
StrikeThrough, Superscript, S
Hidden, SmallCaps, AllCaps,
Position, Kerning, KerningMi
Default, Tab, Font, Bold, Itali
DoubleStrikeThrough, Shado
Outline, Emboss, Engrave, Sc
Animations, CharAccent, For
FontLowAnsi, FontHighAnsi,
CharacterWidthGrid, ColorR
UnderlineColor, PointsBi, Co
FontNameBi, BoldBi, ItalicBi
DiacColor*

wdDialogFormatFont

*Wrap, WidthRule, FixedWidth
HeightRule, FixedHeight,
PositionHorz, PositionHorzR
DistFromText, PositionVert,
PositionVertRel, DistVertFron
MoveWithText, LockAnchor,
RemoveFrame*

wdDialogFormatFrame

*ChapterNumber, NumRestart
NumFormat, StartingNum, L
Separator, DoubleQuote,*

wdDialogFormatPageNumber

wdDialogFormatParagraph

*PgNumberingStyle
LeftIndent, RightIndent, Befo
LineSpacingRule, LineSpacin
Alignment, WidowControl,
KeepWithNext, KeepTogether,
PageBreak, NoLineNum, Dor
Tab, FirstIndent, OutlineLeve
Kinsoku, WordWrap, Overflow
TopLinePunct, AutoSpaceDE,
LineHeightGrid, AutoSpaceD
CharAlign, CharacterUnitLef
AdjustRight, CharacterUnitFi
CharacterUnitRightIndent,
LineUnitBefore, LineUnitAfte
OrientationBi*

wdDialogFormatPicture

*SetSize, CropLeft, CropRight,
CropBottom, ScaleX, ScaleY,
SizeY*

wdDialogFormatRetAddrFonts

*Points, Underline, Color,
StrikeThrough, Superscript, S
Hidden, SmallCaps, AllCaps,
Position, Kerning, KerningMi
Default, Tab, Font, Bold, Itali
DoubleStrikeThrough, Shado
Outline, Emboss, Engrave, Sc
Animations, CharAccent, Fon
FontLowAnsi, FontHighAnsi,
CharacterWidthGrid, ColorR
UnderlineColor, PointsBi, Co
FontNameBi, BoldBi, ItalicBi
DiacColor*

wdDialogFormatSectionLayout

*SectionStart, VertAlign, Endn
LineNum, StartingNum, Fron
CountBy, NumMode, Section'*

wdDialogFormatStyle

*Name, Delete, Merge, NewNa
BasedOn, NextStyle, Type, Fil
Source, AddToTemplate, Defi
Rename, Apply, New*

wdDialogFormatStyleGallery	<i>Template, Preview</i>
wdDialogFormatStylesCustom	(none)
wdDialogFormatTabs	<i>Position, DefTabs, Align, Lea Clear, ClearAll</i>
wdDialogFormatTheme	(none)
wdDialogFormFieldHelp	(none)
wdDialogFormFieldOptions	<i>Entry, Exit, Name, Enable, Te TextWidth, TextDefault, TextF CheckSize, CheckWidth, Chec Type, OwnHelp, HelpText, Ow StatText, Calculate</i>
wdDialogFrameSetProperties	(none)
wdDialogHelpAbout	<i>APPNAME, APPCOPYRIGH APPUSERNAME, APPORGANIZATION, APPSERIALNUMBER</i>
wdDialogHelpWordPerfectHelp	<i>WPCCommand, HelpText, DemoGuidance</i>
wdDialogHelpWordPerfectHelpOptions	<i>CommandKeyHelp, DocNavK MouseSimulation, DemoGuid DemoSpeed, HelpType</i>
wdDialogHorizontalInVertical	(none)
wdDialogIMESetDefault	(none)
wdDialogInsertAddCaption	<i>Name</i>
wdDialogInsertAutoCaption	<i>Clear, ClearAll, Object, Label Name, SortBy, Add, Delete, G Hidden</i>
wdDialogInsertBookmark	<i>Type</i>
wdDialogInsertBreak	<i>Label, TitleAutoText, Title, De Position, AutoCaption</i>
wdDialogInsertCaption	<i>Label, FormatNumber, ChapterNumber, Level, Separ CapNumberingStyle</i>
wdDialogInsertCaptionNumbering	<i>ReferenceType, ReferenceKin ReferenceItem, InsertAsHype InsertPosition</i>
wdDialogInsertCrossReference	

wdDialogInsertDatabase	<i>Format, Style, LinkToSource, Connection, SQLStatement, SQLStatement1, PasswordDot, PasswordDot, DataSource, Fr IncludeFields, WritePassword WritePasswordDot</i>
wdDialogInsertDateTime	<i>DateTimePic, InsertAsField, DbCharField, DateLanguage, CalendarType</i>
wdDialogInsertField	<i>Field</i>
wdDialogInsertFile	<i>Name, Range, ConfirmConve Link, Attachment</i>
wdDialogInsertFootnote	<i>Reference, NoteType, Symbol Entry, Exit, Name, Enable, Te TextWidth, TextDefault, TextF</i>
wdDialogInsertFormField	<i>CheckSize, CheckWidth, Chec Type, OwnHelp, HelpText, Ow StatText, Calculate</i>
wdDialogInsertHyperlink	<i>(none)</i>
wdDialogInsertIndex	<i>Outline, Fields, From, To, Tal AddedStyles, Caption, HeadingSeparator, Replace, MarkEntry, AutoMark, Mark Type, RightAlignPageNumber KeepFormatting, Columns, C Label, ShowPageNumbers, AccentedLetters, Filter, SortB TOCUseHyperlinks, TOCHidePageNumInWeb, IndexLanguage</i>
wdDialogInsertIndexAndTables	<i>Outline, Fields, From, To, Tal AddedStyles, Caption, HeadingSeparator, Replace, MarkEntry, AutoMark, Mark Type, RightAlignPageNumber KeepFormatting, Columns, C Label, ShowPageNumbers,</i>

wdDialogInsertMergeField

wdDialogInsertNumber

wdDialogInsertObject

wdDialogInsertPageNumbers

wdDialogInsertPicture

wdDialogInsertSubdocument

wdDialogInsertSymbol

wdDialogInsertTableOfAuthorities

wdDialogInsertTableOfContents

*AccentedLetters, Filter, SortB
TOCUseHyperlinks,
TOCHidePageNumInWeb,
IndexLanguage*

MergeField, WordField

NumPic

*IconNumber, FileName, Link
DisplayIcon, Tab, Class, Icon.
Caption, Floating*

Type, Position, FirstPage

*Name, LinkToFile, New, Floa
Name, ConfirmConversions, 1*

*LinkToSource, AddToMru,
PasswordDoc, PasswordDot, 1*

*WritePasswordDoc, WritePass
Connection, SQLStatement,
SQLStatement1, Format, Enc
Visible*

*Font, Tab, CharNum, Unicod
Outline, Fields, From, To, Tal
AddedStyles, Caption,*

*HeadingSeparator, Replace,
MarkEntry, AutoMark, Mark*

*Type, RightAlignPageNumber
KeepFormatting, Columns, C*

*Label, ShowPageNumbers,
AccentedLetters, Filter, SortB*

*TOCUseHyperlinks,
TOCHidePageNumInWeb,
IndexLanguage*

*Outline, Fields, From, To, Tal
AddedStyles, Caption,*

*HeadingSeparator, Replace,
MarkEntry, AutoMark, Mark*

*Type, RightAlignPageNumber
KeepFormatting, Columns, C*

Label, ShowPageNumbers,

wdDialogInsertTableOfFigures

AccentedLetters, Filter, SortB, TOCUseHyperlinks, TOCHidePageNumInWeb, IndexLanguage

Outline, Fields, From, To, Table AddedStyles, Caption, HeadingSeparator, Replace, MarkEntry, AutoMark, MarkType, RightAlignPageNumber, KeepFormatting, Columns, ColumnLabel, ShowPageNumbers, AccentedLetters, Filter, SortB, TOCUseHyperlinks, TOCHidePageNumInWeb, IndexLanguage

wdDialogInsertWebComponent

(none)

SenderCity, DateFormat, IncludeHeaderFooter, LetterSize, Letterhead, LetterheadLocation, LetterheadSize, RecipientName, RecipientAddress, Salutation, SalutationType, RecipientGender, RecipientReference,

wdDialogLetterWizard

MailingInstructions, AttentionLetterSubject, CCList, SenderReturnAddress, Closing, SenderJobTitle, SenderCompany, SenderInitials, EnclosureNumber, PageDesign, InfoBlock, SenderReturnAddressSF, RecipientCode, SenderReference, ListType

wdDialogListCommands

CheckErrors, Destination, MergeRecords, From, To, Subject, MailMerge, QueryOptions, MailSubject, MailAsAttachment, MailAddress

wdDialogMailMerge

wdDialogMailMergeCheck

CheckErrors

wdDialogMailMergeCreateDataSource	<i>FileName, PasswordDoc, Pas HeaderRecord, MSQuery, SQLStatement, SQLStatemen Connection, LinkToSource, WritePasswordDoc</i>
wdDialogMailMergeCreateHeaderSource	<i>FileName, PasswordDoc, Pas HeaderRecord, MSQuery, SQLStatement, SQLStatemen Connection, LinkToSource, WritePasswordDoc</i>
wdDialogMailMergeFieldMapping	<i>(none)</i>
wdDialogMailMergeFindRecipient	<i>(none)</i>
wdDialogMailMergeFindRecord	<i>Find, Field</i>
wdDialogMailMergeHelper	<i>Merge, Options</i>
wdDialogMailMergeInsertAddressBlock	<i>(none)</i>
wdDialogMailMergeInsertAsk	<i>Name, Prompt, DefaultBookn AskOnce</i>
wdDialogMailMergeInsertFields	<i>(none)</i>
wdDialogMailMergeInsertFillIn	<i>Prompt, DefaultFillInText, As</i>
wdDialogMailMergeInsertGreetingLine	<i>(none)</i>
wdDialogMailMergeInsertIf	<i>MergeField, Comparison, Coi TrueAutoText, TrueText, FalseAutoText, FalseText</i>
wdDialogMailMergeInsertNextIf	<i>MergeField, Comparison, Coi</i>
wdDialogMailMergeInsertSet	<i>Name, ValueText, ValueAutoT</i>
wdDialogMailMergeInsertSkipIf	<i>MergeField, Comparison, Coi Name, ConfirmConversions, l LinkToSource, AddToMru, PasswordDoc, PasswordDot, l</i>
wdDialogMailMergeOpenDataSource	<i>WritePasswordDoc, WritePass Connection, SQLStatement, SQLStatement1, Format, Enc Visible Name, ConfirmConversions, l LinkToSource, AddToMru, PasswordDoc, PasswordDot, l</i>

wdDialogMailMergeOpenHeaderSource	<i>WritePasswordDoc, WritePass Connection, SQLStatement, SQLStatement1, Format, Enc Visible</i>
wdDialogMailMergeQueryOptions	<i>SQLStatement, SQLStatemen</i>
wdDialogMailMergeRecipients	(none)
wdDialogMailMergeSetDocumentType	(none)
wdDialogMailMergeUseAddressBook	<i>AddressBookType</i>
wdDialogMarkCitation	<i>LongCitation, LongCitationA Category, ShortCitation, Next Mark, MarkAll</i>
wdDialogMarkIndexEntry	<i>MarkAll, Entry, Range, Bold, CrossReference, EntryAutoTe CrossReferenceAutoText, Yon</i>
wdDialogMarkTableOfContentsEntry	<i>Entry, EntryAutoText, TableIc</i>
wdDialogNewToolbar	<i>Name, Context</i>
wdDialogNoteOptions	<i>FootnotesAt, FootNumberAs, FootStartingNum, FootRestar EndnotesAt, EndNumberAs, EndStartingNum, EndRestart FootNumberingStyle, EndNumberingStyle</i>
wdDialogOrganizer	<i>Copy, Delete, Rename, Source Destination, Name, NewName</i>
wdDialogPhoneticGuide	(none)
wdDialogReviewAfmtRevisions	(none)
wdDialogSearch	(none)
wdDialogShowRepairs	(none)
wdDialogTableAutoFormat	<i>HideAutoFit, Preview, Forma Borders, Shading, Font, Color HeadingRows, FirstColumn, i LastColumn</i>
wdDialogTableCellOptions	(none)
wdDialogTableColumnWidth	(none)
wdDialogTableDeleteCells	<i>ShiftCells</i>

wdDialogTableFormatCell	<i>Category</i>
wdDialogTableFormula	<i>Formula, NumFormat</i>
wdDialogTableInsertCells	<i>ShiftCells</i>
wdDialogTableInsertRow	<i>NumRows</i>
wdDialogTableInsertTable	<i>ConvertFrom, NumColumns, NumRows, InitialColWidth, VFormat, Apply, AutoFit, SetDWord8</i>
wdDialogTableOfCaptionsOptions	(none)
wdDialogTableOfContentsOptions	(none)
wdDialogTableProperties	(none)
wdDialogTableRowHeight	(none)
wdDialogTableSort	<i>DontSortHdr, FieldNum, TypFieldNum2, Type2, Order2, FieldNum3, Type3, Order3, SSortColumn, CaseSensitive, SIgnoreHe, Diacritics, IgnoreIKashida, Language</i>
wdDialogTableSplitCells	<i>NumColumns, NumRows, MergeBeforeSplit</i>
wdDialogTableTableOptions	(none)
wdDialogTableToText	<i>ConvertTo, NestedTables</i>
wdDialogTableWrapping	(none)
wdDialogTCSCTranslator	<i>Direction, Varients, Translate</i>
wdDialogTextToTable	<i>ConvertFrom, NumColumns, NumRows, InitialColWidth, VFormat, Apply, AutoFit, SetDWord8</i>
wdDialogToolsAcceptRejectChanges	<i>ShowMarks, HideMarks, WraFindPrevious, FindNext, AcceptRevisions, RejectRevisiAcceptAll, RejectAll</i>
wdDialogToolsAdvancedSettings	<i>Application, Option, Setting, lInitialCaps, SentenceCaps, DCapsLock, ReplaceText, FormReplace, With, Add, Delete,</i>

wdDialogToolsAutoCorrect

*SmartQuotes,
CorrectHangulAndAlphabet,
ConvBrackets, ConvQuotes,
ConvPunct,*

wdDialogToolsAutoCorrectExceptions

*ReplaceTextFromSpellingChe
Tab, Name, AutoAdd, Add, De
Tab*

wdDialogToolsAutoManager

TextSize, Show, Update

wdDialogToolsAutoSummarize

*Replace, Font, CharNum, Typ
FormatOutline, AutoUpdate,
FormatNumber, Punctuation,
Points, Hang, Indent, Remove
DoubleQuote*

wdDialogToolsBulletsNumbers

wdDialogToolsCompareDocuments

Name

wdDialogToolsCreateDirectory

Directory

*ExtractAddress, LabelListInd
LabelIndex, LabelDotMatrix,
LabelTray, LabelAcross, Labe
EnvOmitReturn, EnvReturn,
PrintBarCode, SingleLabel, L
LabelColumn, PrintEnvLabel
AddToDocument, EnvWidth,
EnvHeight, EnvPaperSize, Pr*

wdDialogToolsCreateEnvelope

*UseEnvFeeder, Tab, AddrAut
AddrText, AddrFromLeft,
AddrFromTop, RetAddrFrom.
RetAddrFromTop, LabelTopM
LabelSideMargin, LabelVertP
LabelHorPitch, LabelHeight,
LabelWidth, CustomName,
EnvPaperName, DefaultFace
DefaultOrientation, RetAddrA
ExtractAddress, LabelListInd
LabelIndex, LabelDotMatrix,
LabelTray, LabelAcross, Labe
EnvAddress, EnvOmitReturn,
EnvReturn, PrintBarCode, Si*

wdDialogToolsCreateLabels

*LabelRow, LabelColumn,
PrintEnvLabel, AddToDocum
EnvWidth, EnvHeight, EnvPa
PrintFIMA, UseEnvFeeder, 1
AddrAutoText, AddrText,
AddrFromLeft, AddrFromTop
RetAddrFromLeft, RetAddrFr
LabelTopMargin, LabelSideM
LabelVertPitch, LabelHorPitc
LabelHeight, LabelWidth,
CustomName, RetAddrText,
EnvPaperName, DefaultFace
DefaultOrientation, RetAddrA*

wdDialogToolsCustomize

*KeyCode, KeyCode2, MenuTy
Position, AddAll, Category, N
Menu, AddBelow, MenuText,
Add, Remove, ResetAll,
CommandValue, Context, Tab
KeyCode, KeyCode2, MenuTy
Position, AddAll, Category, N*

wdDialogToolsCustomizeKeyboard

*Menu, AddBelow, MenuText,
Add, Remove, ResetAll,
CommandValue, Context, Tab
Context, Position, MenuType,
MenuText, Menu, Add, Remo
Rename*

wdDialogToolsCustomizeMenuBar

*KeyCode, KeyCode2, MenuTy
Position, AddAll, Category, N
Menu, AddBelow, MenuText,
Add, Remove, ResetAll,
CommandValue, Context, Tab*

wdDialogToolsCustomizeMenus

*ExtractAddress, LabelListInd
LabelIndex, LabelDotMatrix,
LabelTray, LabelAcross, Labe
EnvAddress, EnvOmitReturn,
EnvReturn, PrintBarCode, Si
LabelRow, LabelColumn,
PrintEnvLabel, AddToDocum*

wdDialogToolsEnvelopesAndLabels

*EnvWidth, EnvHeight, EnvPa
PrintFIMA, UseEnvFeeder, 1
AddrAutoText, AddrText,
AddrFromLeft, AddrFromTop
RetAddrFromLeft, RetAddrFr
LabelTopMargin, LabelSideM
LabelVertPitch, LabelHorPitc
LabelHeight, LabelWidth,
CustomName, RetAddrText,
EnvPaperName, DefaultFace
DefaultOrientation, RetAddrA*

wdDialogToolsGrammarSettings

(none)

wdDialogToolsHangulHanjaConversion

(none)

wdDialogToolsHighlightChanges

*MarkRevisions, ViewRevision
PrintRevisions, AcceptAll, Rej*

wdDialogToolsHyphenation

*AutoHyphenation, Hyphenate
HyphenationZone,
LimitConsecutiveHyphens*

wdDialogToolsLanguage

*Language, CheckLanguage, 1
NoProof*

wdDialogToolsMacro

*Name, Run, Edit, Show, Delet
Rename, Description, NewNa
SetDesc*

wdDialogToolsMacroRecord

(This dialog box cannot be call
macro.)

wdDialogToolsManageFields

*FieldName, Add, Remove, Re
NewName*

wdDialogToolsMergeDocuments

Name

wdDialogToolsOptions

Tab

wdDialogToolsOptionsAutoFormat

*ApplyStylesHeadings, ApplySi
ApplyBulletedLists,
ApplyStylesOtherParas, Repla
ReplaceOrdinals, ReplaceFra
ReplaceSymbols,
ReplacePlainTextEmphasis,
ReplaceHyperlinks, PreserveS
PlainTextWordMail, ApplyFir*

wdDialogToolsOptionsAutoFormatAsYouType

*MatchParentheses, ReplaceDi
ReplaceAutoSpaces*

*ApplyStylesHeadings, ApplyB
ApplyTables, ApplyDates,*

ApplyBulletedLists,

ApplyNumberedLists, ApplyF

ApplyClosings, ReplaceQuote

ReplaceOrdinals, ReplaceFra

ReplaceSymbols,

ReplacePlainTextEmphasis,

ReplaceHyperlinks, MatchPa

ReplaceAutoSpaces, ReplaceL

FormatListItemBeginning,

DefineStyles, InsertOvers,

InsertClosings, AutoLetterWiz

ShowOptionsFor, ApplyStyles

ApplySkipList, ApplyStylesOtl

ReplaceBullets, AdjustParaM

AdjustTabsSpaces, AdjustEmp

PreserveStyles

DocViewDir, AddCtrlCopy,

HebDoubleQuote, Numbers, l

BiDirectional, ShowDiac,

DiffDiacColor, Date, Advance

MasterDocDir, OutlineDir,

DiacriticColorVal

Product, Default, NoTabHang

NoSpaceRaiseLower, PrintCo

WrapTrailSpaces, NoColumni

ConvMailMergeEsc,

SuppressSpBfAfterPgBrk,

SuppressTopSpacing,

OrigWordTableRules,

TransparentMetafiles,

ShowBreaksInFrames,

SwapBordersFacingPages,

LeaveBackslashAlone,

ExpandShiftReturn,

wdDialogToolsOptionsBidi

wdDialogToolsOptionsCompatibility

*DontULTrailSpace,
DontBalanceSbDbWidth,
SuppressTopSpacingMac5,
SpacingInWholePoints,
PrintBodyTextBeforeHeader,
NoLeading, NoSpaceForUL,
MWSmallCaps, NoExtraLine.
TruncateFontHeight, SubFon
UsePrinterMetrics, WW6Bord
ExactOnTop, SuppressBottom
WPSpaceWidth, WPJustificat
LineWrapLikeWord6,
SpLayoutLikeWW8,
FtnLayoutLikeWW8,
DontUseHTMLParagraphAu
DontAdjustLineHeightInTabl
ForgetLastTabAlignment,
UseAutospaceForFullWidthA
AlignTablesRowByRow,
LayoutRawTableWidth,
LayoutTableRowsApart,
UseWord97LineBreakingRule
ReplaceSelection, DragAndDi
AutoWordSelection, InsForPc
Overtyping, SmartCutPaste,
AllowAccentedUppercase,
PictureEditor, TabIndent, Bsl
InlineConversion, IMELosing
AllowClickAndTypeMouse,
ClickAndTypeParagraphStyle
AutoKeyBi*

wdDialogToolsOptionsEdit

(none)

wdDialogToolsOptionsEditCopyPaste

wdDialogToolsOptionsFileLocations

Path, Setting

wdDialogToolsOptionsFuzzy

*FuzzyCase, FuzzyByte, Fuzzy.
FuzzySmKana, FuzzyMinus,
FuzzyRepSymbol, FuzzyKanji
FuzzyOldKana, FuzzyLongVo
FuzzyDZ, FuzzyBV, FuzzyTC*

wdDialogToolsOptionsGeneral

*FuzzyHF, FuzzyZJ, FuzzyAY, FuzzyKIKU, FuzzyPunct, Fuz
Paging, WPHelp, WPDoc
BlueScreen, ErrorBeeps, Effe
UpdateLinks, SendMailAttach
RecentFiles, RecentFileCoun
ButtonFieldClicks, ShortMen
RTFInClipboard, ConfirmCo
TipWizardActive, AnimatedCu
VirusProtection, SeparateFon
InterpretHIANSIToDBC,
ExitWithRestoreSession, Asia
PixelsInDialogs, UseCharacte
Draft, Reverse, UpdateFields,
Summary, ShowCodes, Annot
ShowHidden, EnvFeederInstc
WidowControl, DfltTrueType,
UpdateLinks, Background,
DrawingObjects, FormsData,
DefaultTray, PSOverText,
MapPaperSize, FractionalWic
PrOrder1, PrOrder2
CreateBackup, FastSaves,
SummaryPrompt, GlobalDotF
NativePictureFormat, Embed.
FormsData, AutoSave, SaveIn
Password, WritePassword,
RecommendReadOnly, Subse
BackgroundSave, DefaultSav
AddCtrlSave
(none)
(none)
AlwaysSuggest,
SuggestFromMainDictOnly,
IgnoreAllCaps, IgnoreMixedI
ResetIgnoreAll, Type, Custom
CustomDict2, CustomDict3,
CustomDict4, CustomDict5,*

wdDialogToolsOptionsPrint

wdDialogToolsOptionsSave

wdDialogToolsOptionsSecurity

wdDialogToolsOptionsSmartTag

wdDialogToolsOptionsSpellingAndGrammar

*CustomDict6, CustomDict7,
CustomDict8, CustomDict9,
CustomDict10,
AutomaticSpellChecking,
FileNamesEmailAliases, User
AutomaticGrammarChecking
ForegroundGrammar, ShowS
Options, RecheckDocument,
IgnoreAuxFind,
IgnoreMissDictSearch,
HideGrammarErrors, CheckS
GrLidUI, SpLidUI, DictLang1
DictLang2, DictLang3, DictLa
DictLang5, DictLang6, DictLa
DictLang8, DictLang9, DictLa
HideSpellingErrors, HebSpell
InitialAlefHamza, FinalYaa,
GermanPostReformSpell, Ara
ProcessCompoundNoun*

wdDialogToolsOptionsTrackChanges

*InsertedTextMark, InsertedTe
DeletedTextMark, DeletedTex
RevisedLinesMark, RevisedLi
HighlightColor,
RevisedPropertiesMark,
RevisedPropertiesColor*

wdDialogToolsOptionsTypography

*KerningPairs, Justification,
PunctLevel, FollowingPunct,
LeadingPunct, ApplyToTempl
JapaneseKinsokuStrict,
FarEastLineBreakLanguage*

wdDialogToolsOptionsUserInfo

Name, Initials, Address

*DraftFont, WrapToWindow,
PicturePlaceHolders, FieldCo
BookMarks, FieldShading, St
HScroll, VScroll, StyleAreaW
Spaces, Paras, Hyphens, Hida
ShowAll, Drawings, Anchors,
TextBoundaries, VRuler, High*

wdDialogToolsOptionsView

wdDialogToolsProtectDocument

wdDialogToolsProtectSection

wdDialogToolsRevisions

wdDialogToolsSpellingAndGrammar

wdDialogToolsTemplates

wdDialogToolsThesaurus

wdDialogToolsUnprotectDocument

wdDialogToolsWordCount

wdDialogTwoLinesInOne

wdDialogUpdateTOC

wdDialogViewZoom

wdDialogWebOptions

wdDialogWindowActivate

*ShowAnimation, ScrnTp, Left
RRuler, OptionalBreak,
EnlargeFontsLessThan,
BrowseToWindow*

*DocumentPassword, NoReset,
Protect, Section*

*MarkRevisions, ViewRevision
PrintRevisions, AcceptAll, Rej*

*SuggestionListBox,
ForegroundGrammar*

Store, Template, LinkStyles

(none)

DocumentPassword

*CountFootnotes, Pages, Word
Characters, DBCs, SBCs,
CharactersIncludingSpaces,
Paragraphs, Lines*

(none)

(none)

*AutoFit, TwoPages, FullPage
NumColumns, NumRows,
ZoomPercent, TextFit*

(none)

Window



OLE Programmatic Identifiers

You can use an OLE programmatic identifier (sometimes called a ProgID) to create an Automation object. The following tables list OLE programmatic identifiers for ActiveX controls, Microsoft Office applications, and Microsoft Office Web Components.

[ActiveX Controls](#)

[Microsoft Access](#)

[Microsoft Excel](#)

[Microsoft Graph](#)

[Microsoft Office Web Components](#)

[Microsoft Outlook](#)

[Microsoft PowerPoint](#)

[Microsoft Word](#)

ActiveX Controls

To create the ActiveX controls listed in the following table, use the corresponding OLE programmatic identifier.

To create this control	Use this identifier
CheckBox	Forms.CheckBox.1
ComboBox	Forms.ComboBox.1
CommandButton	Forms.CommandButton.1
Frame	Forms.Frame.1
Image	Forms.Image.1
Label	Forms.Label.1
ListBox	Forms.ListBox.1
MultiPage	Forms.MultiPage.1
OptionButton	Forms.OptionButton.1
ScrollBar	Forms.ScrollBar.1
SpinButton	Forms.SpinButton.1
TabStrip	Forms.TabStrip.1
TextBox	Forms.TextBox.1
ToggleButton	Forms.ToggleButton.1

Microsoft Access

To create the Microsoft Access objects listed in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Access available on the machine where the macro is running.

To create this object	Use one of these identifiers
Application	Access.Application
CurrentData	Access.CodeData, Access.CurrentData
CurrentProject	Access.CodeProject, Access.CurrentProject
DefaultWebOptions	Access.DefaultWebOptions

Microsoft Excel

To create the Microsoft Excel objects listed in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Excel available on the machine where the macro is running.

To create this object	Use one of these identifiers	Comments
Application	Excel.Application	
Workbook	Excel.AddIn	
Workbook	Excel.Chart	Returns a workbook containing two worksheets; one for the chart and one for its data. The chart worksheet is the active worksheet.
Workbook	Excel.Sheet	Returns a workbook with one worksheet.

Microsoft Graph

To create the Microsoft Graph objects listed in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Graph available on the machine where the macro is running.

To create this object	Use one of these identifiers
Application	MSGraph.Application
Chart	MSGraph.Chart

Microsoft Office Web Components

To create the Microsoft Office Web Components objects listed in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Microsoft Office Web Components available on the machine where the macro is running.

To create this object	Use one of these identifiers
ChartSpace	OWC.Chart
DataSourceControl	OWC.DataSourceControl
ExpandControl	OWC.ExpandControl
PivotTable	OWC.PivotTable
RecordNavigationControl	OWC.RecordNavigationControl
Spreadsheet	OWC.Spreadsheet

Microsoft Outlook

To create the Microsoft Outlook object given in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Outlook available on the machine where the macro is running.

To create this object	Use one of these identifiers
Application	Outlook.Application

Microsoft PowerPoint

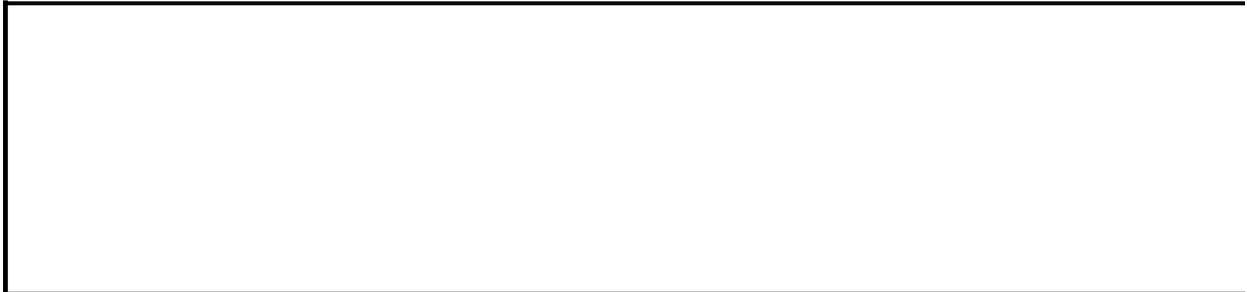
To create the Microsoft PowerPoint object given in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of PowerPoint available on the machine where the macro is running.

To create this object	Use one of these identifiers
Application	PowerPoint.Application

Microsoft Word

To create the Microsoft Word objects listed in the following table, use one of the corresponding OLE programmatic identifiers. If you use an identifier without a version number suffix, you create an object in the most recent version of Word available on the machine where the macro is running.

To create this object	Use one of these identifiers
Application	Word.Application
Document	Word.Document, Word.Template
Global	Word.Global



AddIn Object

[Application](#) | [AddIns \(AddIn\)](#)

Represents a single add-in, either installed or not installed. The **AddIn** object is a member of the [AddIns](#) collection. The **AddIns** collection contains all the add-ins available to Word, regardless of whether or not they're currently loaded. The **AddIns** collection includes global templates or Word add-in libraries (WLLs) displayed in the **Templates and Add-ins** dialog box (**Tools** menu).

Using the AddIn Object

Use **AddIns**(*index*), where *index* is the add-in name or index number, to return a single **AddIn** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown in the **Templates and Add-Ins** dialog box. The following example loads the Letter.dot template as a global template.

```
AddIns("Letter.dot").Installed = True
```

The index number represents the position of the add-in in the list of add-ins in the **Templates and Add-ins** dialog box. The following instruction displays the path of the first available add-in.

```
If Addins.Count >= 1 Then MsgBox Addins(1).Path
```

The following example creates a list of add-ins at the beginning of the active document. The list contains the name, path, and installed state of each available add-in.

```
With ActiveDocument.Range(Start:=0, End:=0)
    .InsertAfter "Name" & vbTab & "Path" & vbTab & "Installed"
    .InsertParagraphAfter
    For Each oAddIn In AddIns
        .InsertAfter oAddIn.Name & vbTab & oAddIn.Path & vbTab _
            & oAddIn.Installed
        .InsertParagraphAfter
    Next oAddIn
    .ConvertToTable
End With
```

Use the [Add](#) method to add an add-in to the list of available add-ins and (optionally) install it using the **Install** argument.

```
AddIns.Add FileName:="C:\Templates\Other\Letter.dot", Install:=True
```

To install an add-in shown in the list of available add-ins, use the [Installed](#) property.

```
AddIns("Letter.dot").Installed = True
```

Note Use the [Compiled](#) property to determine whether an **AddIn** object is a template or a WLL.



AddIns Collection Object

[Application](#)  [AddIns \(AddIn\)](#)

A collection of [AddIn](#) objects that represents all the add-ins available to Word, regardless of whether or not they're currently loaded. The **AddIns** collection includes global templates or Word add-in libraries (WLLs) displayed in the **Templates and Add-ins** dialog box (**Tools** menu).

Using the AddIns Collection

Use the **AddIns** property to return the **AddIns** collection. The following example displays the name and the installed state of each available add-in.

```
For Each ad In AddIns
    If ad.Installed = True Then
        MsgBox ad.Name & " is installed"
    Else
        MsgBox ad.Name & " is available but not installed"
    End If
Next ad
```

Use the [Add](#) method to add an add-in to the list of available add-ins and (optionally) install it using the **Install** argument.

```
AddIns.Add FileName:="C:\Templates\Other\Letter.dot", Install:=True
```

To install an add-in shown in the list of available add-ins, use the **Installed** property.

```
AddIns("Letter.dot").Installed = True
```

Use **AddIns(index)**, where *index* is the add-in name or index number, to return a single **AddIn** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown in the **Templates and Add-ins** dialog box. To install an add-in shown in the list of available add-ins, use the [Installed](#) property. The following example loads the Letter.dot template as a global template.

```
AddIns("Letter.dot").Installed = True
```

Note If the add-in is not located in the User Templates, Workgroup Templates, or Startup folder, you must specify the full path and file name when indexing an add-in by name.

Remarks

Use the [Compiled](#) property to determine whether an **AddIn** object is a template or a WLL.

--

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.

Note Because each adjustable shape has a different set of adjustments, the best way to verify the adjustment behavior for a specific shape is to manually create an instance of the shape, make adjustments with the macro recorder turned on, and then examine the recorded code.

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 half way 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.

```
Set rac = ActiveDocument.Shapes _  
    .AddShape(msoShapeRightArrowCallout, 10, 10, 250, 190)  
With rac.Adjustments  
    .Item(1) = 0.5    'adjusts width of text box  
    .Item(2) = 0.15  'adjusts width of arrow head  
    .Item(3) = 0.8    'adjusts length of arrow head  
    .Item(4) = 0.4    'adjusts width of arrow neck  
End With
```



Application Object

[Application](#) └ Multiple objects

Represents the Microsoft Word 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 user name for Word.

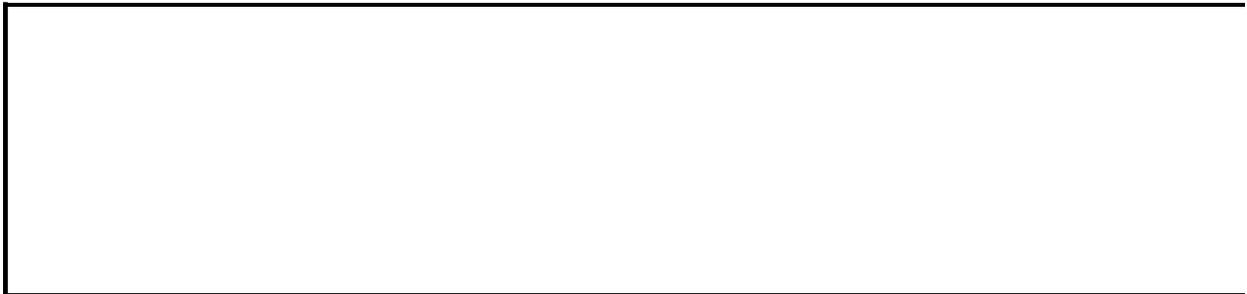
```
MsgBox Application.UserName
```

Many of the properties and methods that return the most common user-interface objects — such as the active document (**ActiveDocument** property) — can be used without the **Application** object qualifier. For example, instead of writing `Application.ActiveDocument.PrintOut`, you can write `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.

Remarks

To use Automation (formerly OLE Automation) to control Word from another application, use Visual Basic's **CreateObject** or **GetObject** function to return a Word **Application** object. The following Microsoft Excel example starts Word (if it's not already running) and opens an existing document.

```
Set wrd = GetObject(, "Word.Application")  
wrd.Visible = True  
wrd.Documents.Open "C:\My Documents\Temp.doc"  
Set wrd = Nothing
```



AutoCaption Object

[Application](#) | [AutoCaptions \(AutoCaption\)](#)

Represents a single caption that can be automatically added when items such as tables, pictures, or OLE objects are inserted into a document. The **AutoCaption** object is a member of the [AutoCaptions](#) collection. The **AutoCaptions** collection contains all the captions listed in the **AutoCaption** dialog box (**Insert** menu).

Using the AutoCaption Object

Use **AutoCaptions**(*index*), where *index* is the caption name or index number, to return a single **AutoCaption** object. The caption names correspond to the items listed in the **AutoCaption** dialog box (**Insert** menu). You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown in the **AutoCaption** dialog box. The following example enables autocaptions for Word tables.

```
AutoCaptions("Microsoft Word Table").AutoInsert = True
```

The index number represents the position of the **AutoCaption** object in the list of items in the **AutoCaption** dialog box. The following example displays the name of the first item listed in the **AutoCaption** dialog box.

```
MsgBox AutoCaptions(1).Name
```

AutoCaption objects cannot be programmatically added to or deleted from the **AutoCaptions** collection.



AutoCaptions Collection Object

[Application](#) | [AutoCaptions \(AutoCaption\)](#)

A collection of [AutoCaption](#) objects that represent the captions that can be automatically added when items such as tables, pictures, or OLE objects are inserted into a document.

Using the AutoCaptions Collection

Use the **AutoCaptions** property to return the **AutoCaptions** collection. The following example displays the names of the selected items in the **AutoCaption** dialog box.

```
For Each autoCap In AutoCaptions
    If autoCap.AutoInsert = True Then
        MsgBox autoCap.Name & " is configured for auto insert"
    End If
Next autoCap
```

The **AutoCaptions** collection contains all the captions listed in the **AutoCaption** dialog box (**Insert** menu). **AutoCaption** objects cannot be programmatically added to or deleted from the **AutoCaptions** collection.

Use **AutoCaptions(index)**, where *index* is the caption name or index number, to return a single **AutoCaption** object. The caption names correspond to the items listed in the **AutoCaption** dialog box (**Insert** menu). You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown in the **AutoCaption** dialog box. The following example displays the caption text "Microsoft Word Table."

```
MsgBox AutoCaptions("Microsoft Word Table").CaptionLabel.Name
```

The index number represents the position of the **AutoCaption** object in the list of captions in the **AutoCaption** dialog box. The following example displays the name of the first item selected in the **AutoCaption** dialog box.

```
MsgBox AutoCaptions(1).Name
```



AutoCorrect Object

[Application](#) └ [AutoCorrect](#)
└ Multiple objects

Represents the AutoCorrect functionality in Word.

Using the AutoCorrect Object

Use the **AutoCorrect** property to return the **AutoCorrect** object. The following example enables the AutoCorrect options and creates an AutoCorrect entry.

```
With AutoCorrect
    .CorrectCapsLock = True
    .CorrectDays = True
    .Entries.Add Name:="usualy", Value:="usually"
End With
```

The **Entries** property returns the **AutoCorrectEntries** object that represents the AutoCorrect entries in the **AutoCorrect** dialog box (**Tools** menu).



AutoCorrectEntries Collection Object

[Application](#) └ [AutoCorrect](#)
└ [AutoCorrectEntries \(AutoCorrectEntry\)](#)

A collection of [AutoCorrectEntry](#) objects that represent all the AutoCorrect entries available to Word. The **AutoCorrectEntries** collection includes all the entries in the **AutoCorrect** dialog box (**Tools** menu).

Using the AutoCorrectEntries Collection

Use the **Entries** property to return the **AutoCorrectEntries** collection. The following example displays the number of **AutoCorrectEntry** objects in the **AutoCorrectEntries** collection.

```
MsgBox AutoCorrect.Entries.Count
```

Use the [Add](#) or the [AddRichText](#) method to add an AutoCorrect entry to the list of available entries. The following example adds a plain-text AutoCorrect entry for the misspelling of the word "their."

```
AutoCorrect.Entries.Add Name:="thier", Value:="their"
```

The following example creates an AutoCorrect entry named "PMO" based on the text and formatting of the selection.

```
AutoCorrect.Entries.AddRichText Name:="PMO", Range:=Selection.Range
```

Use **Entries(index)**, where *index* is the AutoCorrect entry name or index number, to return a single **AutoCorrectEntry** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown under **Replace** in the **AutoCorrect** dialog box. The following example sets the value of an existing AutoCorrect entry named "teh."

```
AutoCorrect.Entries("teh").Value = "the"
```

The following example displays the name and value of the first AutoCorrect entry.

```
MsgBox "Name = " & AutoCorrect.Entries(1).Name & vbCr & _  
      "Value " & AutoCorrect.Entries(1).Value
```



AutoCorrectEntry Object

[Application](#) └ [AutoCorrect](#)
└ [AutoCorrectEntries \(AutoCorrectEntry\)](#)

Represents a single AutoCorrect entry. The **AutoCorrectEntry** object is a member of the **AutoCorrectEntries** collection. The **AutoCorrectEntries** collection includes the entries in the **AutoCorrect** dialog box (**Tools** menu).

Using the **AutoCorrectEntry** Object

Use **Entries**(*index*), where *index* is the AutoCorrect entry name or index number, to return a single **AutoCorrectEntry** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown under **Replace** in the **AutoCorrect** dialog box. The following example sets the value of the AutoCorrect entry named "teh."

```
AutoCorrect.Entries("teh").Value = "the"
```

Use the [Apply](#) method to insert an AutoCorrect entry at the specified range. The following example adds an AutoCorrect entry and then inserts it in place of the selection.

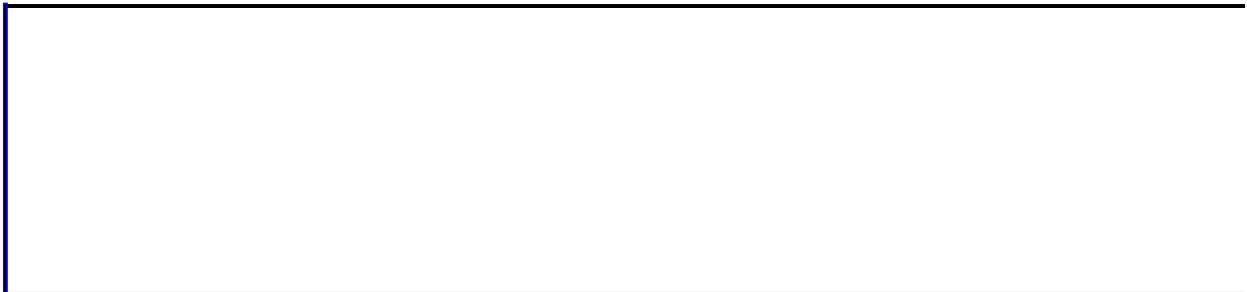
```
AutoCorrect.Entries.Add Name:="hellp", Value:="hello"  
AutoCorrect.Entries("hellp").Apply Range:=Selection.Range
```

Use either the [Add](#) or [AddRichText](#) method to add an AutoCorrect entry to the list of available entries. The following example adds a plain-text AutoCorrect entry for the misspelling of the word "their."

```
AutoCorrect.Entries.Add Name:="thier", Value:="their"
```

The following example creates an AutoCorrect entry named "PMO" based on the text and formatting of the selection.

```
AutoCorrect.Entries.AddRichText Name:="PMO", Range:=Selection.Range
```



AutoTextEntries Collection Object

[Application](#) └ [Templates \(Template\)](#)
└ [AutoTextEntries \(AutoTextEntry\)](#)

A collection of [AutoTextEntry](#) objects that represent the AutoText entries in a template. The **AutoTextEntries** collection includes all the entries listed on the **AutoText** tab in the **AutoCorrect** dialog box (**Tools** menu).

Using the **AutoTextEntries** Object

Use the **AutoTextEntries** property to return the **AutoTextEntries** collection. The following example determines whether an **AutoTextEntry** object named "test" is in the **AutoTextEntries** collection.

```
For Each i In NormalTemplate.AutoTextEntries
    If LCase(i.Name) = "test" Then MsgBox "AutoText entry exists"
Next i
```

Use the [Add](#) method to add an AutoText entry to the **AutoTextEntries** collection. The following example adds an AutoText entry named "Blue" based on the text of the selection.

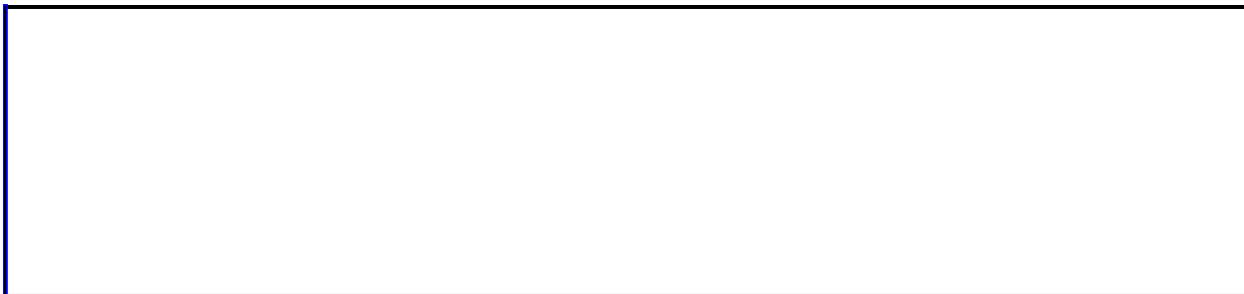
```
NormalTemplate.AutoTextEntries.Add Name:="Blue", _
    Range:=Selection.Range
```

Use **AutoTextEntries(index)**, where *index* is the AutoText entry name or index number, to return a single **AutoTextEntry** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown on the **AutoText** tab in the **AutoCorrect** dialog box. The following example sets the value of an existing AutoText entry named "cName."

```
NormalTemplate.AutoTextEntries("cName").Value = _
    "The Johnson Company"
```

The following example displays the name and value of the first AutoText entry in the template attached to the active document.

```
Set myTemplate = ActiveDocument.AttachedTemplate
MsgBox "Name = " & myTemplate.AutoTextEntries(1).Name & vbCr _
    & "Value " & myTemplate.AutoTextEntries(1).Value
```



AutoTextEntry Object

[Application](#) └ [Templates \(Template\)](#)
└ [AutoTextEntries \(AutoTextEntry\)](#)

Represents a single AutoText entry. The **AutoTextEntry** object is a member of the [AutoTextEntries](#) collection. The **AutoTextEntries** collection contains all the AutoText entries in the specified template. The entries are listed on the **AutoText** tab in the **AutoCorrect** dialog box (**Tools** menu).

Using the AutoTextEntry Object

Use **AutoTextEntries(index)**, where *index* is the AutoText entry name or index number, to return a single **AutoTextEntry** object. You must exactly match the spelling (but not necessarily the capitalization) of the name, as it's shown on the **AutoText** tab in the **AutoCorrect** dialog box. The following example sets the value of an existing AutoText entry named "cName."

```
NormalTemplate.AutoTextEntries("cName").Value = _  
    "The Johnson Company"
```

The following example displays the name and value of the first AutoText entry in the template attached to the active document.

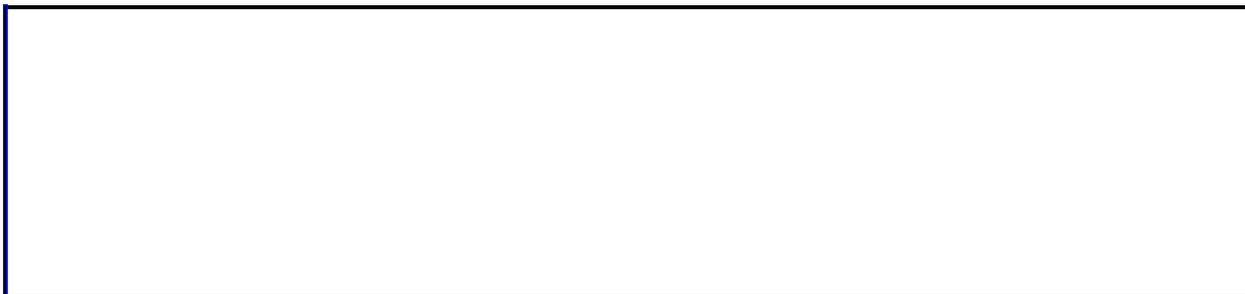
```
Set myTemplate = ActiveDocument.AttachedTemplate  
MsgBox "Name = " & myTemplate.AutoTextEntries(1).Name & vbCr _  
    & "Value " & myTemplate.AutoTextEntries(1).Value
```

The following example inserts the global AutoText entry named "TheWorld" at the insertion point.

```
Selection.Collapse Direction:=wdCollapseEnd  
NormalTemplate.AutoTextEntries("TheWorld").Insert _  
    Where:=Selection.Range
```

Use the [Add](#) method to add an **AutoTextEntry** object to the **AutoTextEntries** collection. The following example adds an AutoText entry named "Blue" based on the text of the selection.

```
NormalTemplate.AutoTextEntries.Add Name:="Blue", _  
    Range:=Selection.Range
```



Bookmark Object

Multiple objects [Bookmarks \(Bookmark\)](#)
[Range](#)

Represents a single bookmark. The **Bookmark** object is a member of the [Bookmarks](#) collection. The **Bookmarks** collection includes all the bookmarks listed in the **Bookmark** dialog box (**Insert** menu).

Using the Bookmark Object

Use **Bookmarks**(*index*), where *index* is the bookmark name or index number, to return a single **Bookmark** object. You must exactly match the spelling (but not necessarily the capitalization) of the bookmark name. The following example selects the bookmark named "temp" in the active document.

```
ActiveDocument.Bookmarks("temp").Select
```

The index number represents the position of the bookmark in the **Selection** or **Range** object. For the **Document** object, the index number represents the position of the bookmark in the alphabetic list of bookmarks in the **Bookmarks** dialog box (click **Name** to sort the list of bookmarks alphabetically). The following example displays the name of the second bookmark in the **Bookmarks** collection.

```
MsgBox ActiveDocument.Bookmarks(2).Name
```

Use the [Add](#) method to add a bookmark to a document range. The following example marks the selection by adding a bookmark named "temp."

```
ActiveDocument.Bookmarks.Add Name:="temp", Range:=Selection.Range
```

Remarks

Use the [BookmarkID](#) property with a range or selection object to return the index number of the **Bookmark** object in the **Bookmarks** collection. The following example displays the index number of the bookmark named "temp" in the active document.

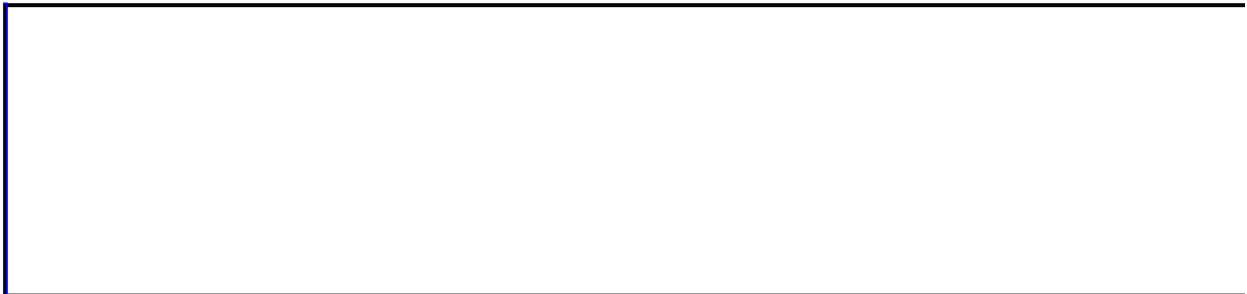
```
MsgBox ActiveDocument.Bookmarks("temp").Range.BookmarkID
```

You can use [predefined bookmarks](#) with the **Bookmarks** property. The following example sets the bookmark named "currpara" to the location marked by the predefined bookmark named "\Para".

```
ActiveDocument.Bookmarks("\Para").Copy "currpara"
```

Use the [Exists](#) method to determine whether a bookmark already exists in the selection, range, or document. The following example ensures that the bookmark named "temp" exists in the active document before selecting the bookmark.

```
If ActiveDocument.Bookmarks.Exists("temp") = True Then  
    ActiveDocument.Bookmarks("temp").Select  
End If
```



Bookmarks Collection Object

Multiple objects [↳ Bookmarks \(Bookmark\)](#)
[↳ Range](#)

A collection of [Bookmark](#) objects that represent the bookmarks in the specified selection, range, or document.

Using the Bookmarks Collection

Use the **Bookmarks** property to return the **Bookmarks** collection. The following example ensures that the bookmark named "temp" exists in the active document before selecting the bookmark.

```
If ActiveDocument.Bookmarks.Exists("temp") = True Then
    ActiveDocument.Bookmarks("temp").Select
End If
```

Use the [Add](#) method to set a bookmark for a range in a document. The following example marks the selection by adding a bookmark named "temp".

```
ActiveDocument.Bookmarks.Add Name:="temp", Range:=Selection.Range
```

Use **Bookmarks(index)**, where *index* is the bookmark name or index number, to return a single **Bookmark** object. You must exactly match the spelling (but not necessarily the capitalization) of the bookmark name. The following example selects the bookmark named "temp" in the active document.

```
ActiveDocument.Bookmarks("temp").Select
```

The index number represents the position of the bookmark in the **Selection** or **Range** object. For the **Document** object, the index number represents the position of the bookmark in the alphabetic list of bookmarks in the **Bookmarks** dialog box (click **Name** to sort the list of bookmarks alphabetically). The following example displays the name of the second bookmark in the **Bookmarks** collection.

```
MsgBox ActiveDocument.Bookmarks(2).Name
```

Remarks

The [ShowHidden](#) property effects the number of elements in the **Bookmarks** collection. If **ShowHidden** is **True**, hidden bookmarks are included in the **Bookmarks** collection.

Border Object

Multiple objects [└ Borders \(LineFormat\)](#)

Represents a border of an object. The **Border** object is a member of the [Borders](#) collection.

Using the Border Object

Use **Borders**(*index*), where *index* identifies the border, to return a single **Border** object. *Index* can be one of the following **WdBorderType** constants: **wdBorderBottom**, **wdBorderDiagonalDown**, **wdBorderDiagonalUp**, **wdBorderHorizontal**, **wdBorderLeft**, **wdBorderRight**, **wdBorderTop**, or **wdBorderVertical**. Use the [LineStyle](#) property to apply a border line to a **Border** object. The following example applies a double-line border below the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).Borders(wdBorderBottom)
    .LineStyle = wdLineStyleDouble
    .LineWidth = wdLineWidth025pt
End With
```

The following example applies a single-line border around the first character in the selection.

```
With Selection.Characters(1)
    .Font.Size = 36
    .Borders.Enable = True
End With
```

The following example adds an art border around each page in the first section.

```
For Each aBorder In ActiveDocument.Sections(1).Borders
    With aBorder
        .ArtStyle = wdArtSeattle
        .ArtWidth = 20
    End With
Next aBorder
```

Border objects cannot be added to the **Borders** collection. The number of members in the **Borders** collection is finite and varies depending on the type of object. For example, a table has six elements in the **Borders** collection, whereas a paragraph has four.



Borders Collection Object

Multiple objects [└ Borders \(Border\)](#)

A collection of [Border](#) objects that represent the borders of an object.

Using the Borders Collection

Use the **Borders** property to return the **Borders** collection. The following example applies the default border around the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Borders.Enable = True
```

Border objects cannot be added to the **Borders** collection. The number of members in the **Borders** collection is finite and varies depending on the type of object. For example, a table has six elements in the **Borders** collection, whereas a paragraph has four.

Use **Borders(index)**, where *index* identifies the border, to return a single **Border** object. *Index* can be one of the following **WdBorderType** constants: **wdBorderBottom**, **wdBorderDiagonalDown**, **wdBorderDiagonalUp**, **wdBorderHorizontal**, **wdBorderLeft**, **wdBorderRight**, **wdBorderTop**, or **wdBorderVertical**. Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed. Use the [LineStyle](#) property to apply a border line to a **Border** object. The following example applies a double-line border below the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).Borders(wdBorderBottom)
    .LineStyle = wdLineStyleDouble
    .LineWidth = wdLineWidth025pt
End With
```

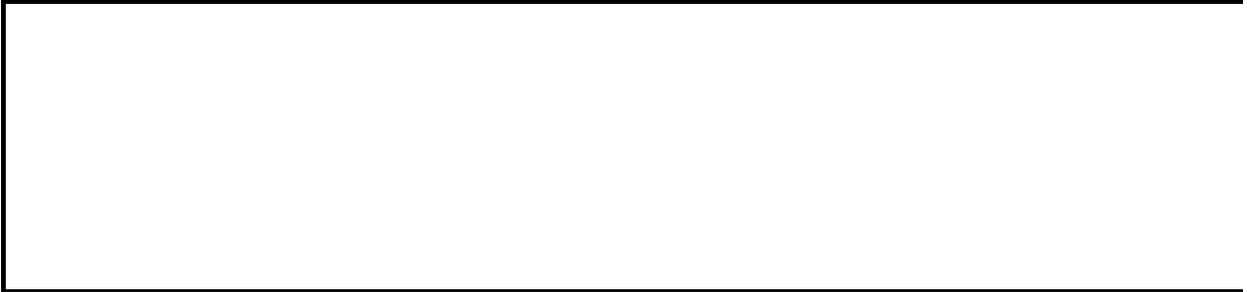
The following example applies a single-line border around the first character in the selection.

```
With Selection.Characters(1)
    .Font.Size = 36
    .Borders.Enable = True
End With
```

The following example adds an art border around each page in the first section.

```
For Each aBorder In ActiveDocument.Sections(1).Borders
    With aBorder
```

```
        .ArtStyle = wdArtSeattle
        .ArtWidth = 20
    End With
Next aBorder
```



Browser Object

[Application](#)  [Browser](#)

Represents the browser tool used to move the insertion point to objects in a document. This tool is comprised of the three buttons at the bottom of the vertical scroll bar.

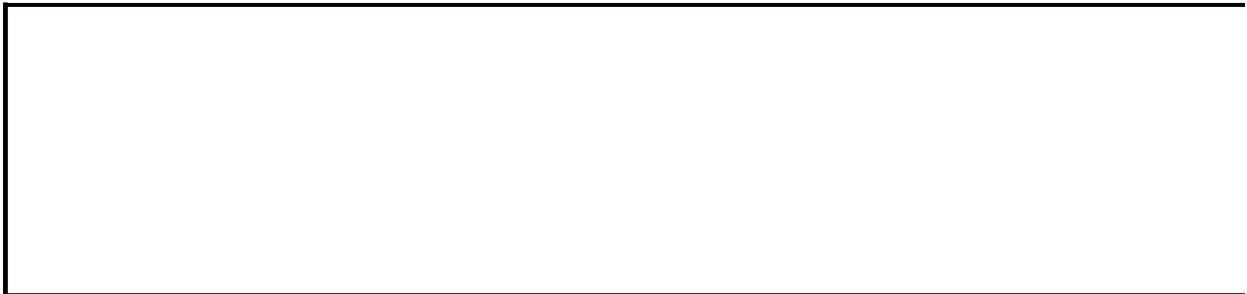
Using the Browser Object

Use the **Browser** property to return the **Browser** object. The following example moves the insertion point just before the next field in the active document.

```
With Application.Browser
    .Target = wdBrowseField
    .Next
End With
```

The following example moves the insertion point to the previous table and selects it.

```
With Application.Browser
    .Target = wdBrowseTable
    .Previous
End With
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).Select
End If
```



CalloutFormat Object

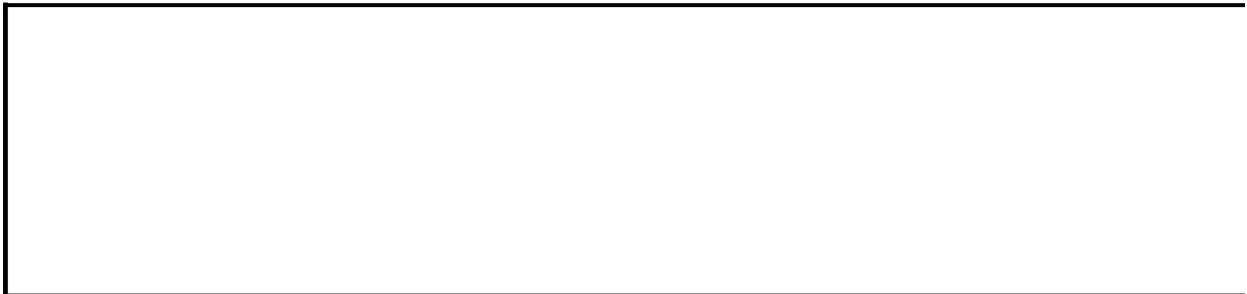
[Shapes \(Shape\)](#) └ [CalloutFormat](#)

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 specifies the following attributes of shape three (a line callout) on the active document: the callout will have a vertical accent bar that separates the text from the callout line; the angle between the callout line and the side of the callout text box will be 30 degrees; there will be no border around the callout text; the callout line will be attached to the top of the callout text box; and the callout line will contain two segments. For this example to work, shape three must be a callout.

```
With ActiveDocument.Shapes(3).Callout
    .Accent = True
    .Angle = msoCalloutAngle30
    .Border = False
    .PresetDrop msoCalloutDropTop
    .Type = msoCalloutThree
End With
```



CanvasShapes Collection

Multiple objects [↳CanvasShapes](#)
↳Multiple objects

Represents the shapes in a drawing canvas.

Using the CanvasShapes collection

Use the [CanvasItems](#) property of either a [Shape](#) or [ShapeRange](#) object to return a **CanvasShapes** collection. To add shapes to a drawing canvas, use the following methods of the **CanvasShapes** collection: [AddCallout](#), [AddConnector](#), [AddCurve](#), [AddLabel](#), [AddLine](#), [AddPicture](#), [AddPolyline](#), [AddShape](#), [AddTextbox](#), [AddTextEffect](#), or [BuildFreeForm](#). The following example adds a drawing canvas to the active document and then adds three shapes to the drawing canvas.

```
Sub AddCanvasShapes()  
    Dim shpCanvas As Shape  
    Dim shpCanvasShapes As CanvasShapes  
    Dim shpCnvItem As Shape  
  
    'Adds a new canvas to the document  
    Set shpCanvas = ActiveDocument.Shapes _  
        .AddCanvas(Left:=100, Top:=75, _  
            Width:=50, Height:=75)  
    Set shpCanvasShapes = shpCanvas.CanvasItems  
  
    'Adds shapes to the CanvasShapes collection  
    With shpCanvasShapes  
        .AddShape Type:=msoShapeRectangle, _  
            Left:=0, Top:=0, Width:=50, Height:=50  
        .AddShape Type:=msoShapeOval, _  
            Left:=5, Top:=5, Width:=40, Height:=40  
        .AddShape Type:=msoShapeIsoscelesTriangle, _  
            Left:=0, Top:=25, Width:=50, Height:=50  
    End With  
End Sub
```

Use [CanvasItems\(index\)](#), where *index* is the name or the index number, to return a single shape in the **CanvasShapes** collection. The following example sets the [Line](#) and [Fill](#) properties and vertically flips the third shape in a drawing canvas.

```
Sub CanvasShapeThree()  
    With ActiveDocument.Shapes(1).CanvasItems(3)  
        .Line.ForeColor.RGB = RGB(50, 0, 255)  
        .Fill.ForeColor.RGB = RGB(50, 0, 255)  
        .Flip msoFlipVertical  
    End With  
End Sub
```

```
End With  
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. Use the [Name](#) property to reference the default name or to assign a more meaningful name to a shape.



CaptionLabel Object

[Application](#) | [CaptionLabels \(CaptionLabel\)](#)

Represents a single caption label. The **CaptionLabel** object is a member of the [CaptionLabels](#) collection. The items in the **CaptionLabels** collection are listed in the **Label** box in the **Caption** dialog box (**Insert** menu).

Using the CaptionLabel Object

Use **CaptionLabels**(*index*), where *index* is the caption label name or index number, to return a single **CaptionLabel** object. The following example sets the numbering style for the Figure caption label.

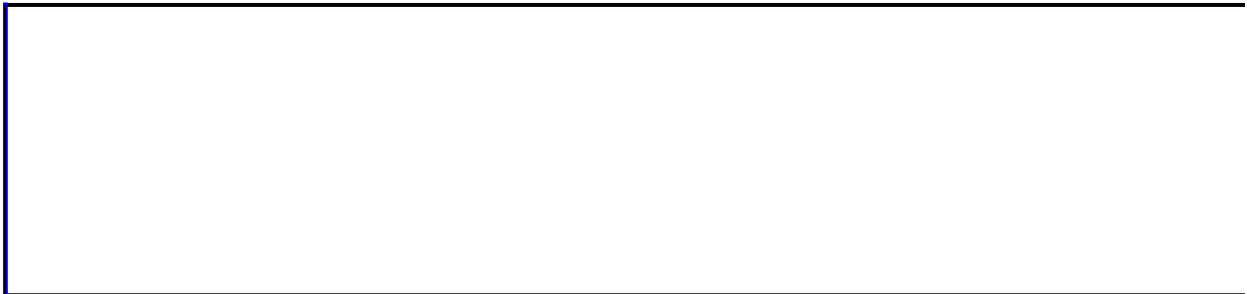
```
CaptionLabels("Figure").NumberStyle = _  
    wdCaptionNumberStyleLowercaseLetter
```

The index number represents the position of the caption label in the **CaptionLabels** collection. The following example displays the first caption label.

```
MsgBox CaptionLabels(1).Name
```

Use the [Add](#) method to add a custom caption label. The following example adds a caption label named "Photo."

```
CaptionLabels.Add Name:="Photo"
```



CaptionLabels Collection Object

[Application](#) | [CaptionLabels \(CaptionLabel\)](#)

A collection of **CaptionLabel** objects that represent the available caption labels. The items in the **CaptionLabels** collection are listed in the **Label** box in the **Caption** dialog box (**Insert** menu).

Using the CaptionLabels Collection

Use the **CaptionLabels** property to return the **CaptionLabels** collection. By default, the **CaptionLabels** collection includes the three built-in caption labels: Figure, Table, and Equation.

Use the [Add](#) method to add a custom caption label. The following example adds a caption label named "Photo."

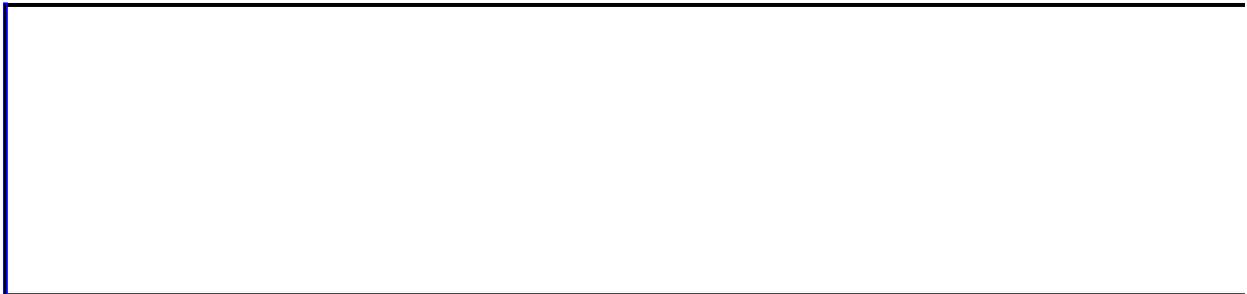
```
CaptionLabels.Add Name:="Photo"
```

Use **CaptionLabels(index)**, where *index* is the caption label name or index number, to return a single **CaptionLabel** object. The following example sets the numbering style for the Figure caption label.

```
CaptionLabels("Figure").NumberStyle = _  
    wdCaptionNumberStyleLowercaseLetter
```

The index number represents the position of the caption label in the **CaptionLabels** collection. The following example displays the first caption label.

```
MsgBox CaptionLabels(1).Name
```



Cell Object

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

Represents a single table cell. The **Cell** object is a member of the [Cells](#) collection. The **Cells** collection represents all the cells in the specified object.

Using the Cell Object

Use **Cell**(*row*, *column*), where *row* is the row number and *column* is the column number, or **Cells**(*index*), where *index* is the index number, to return a **Cell** object. The following example applies shading to the second cell in the first row.

```
Set myCell = ActiveDocument.Tables(1).Cell(Row:=1, Column:=2)
myCell.Shading.Texture = wdTexture20Percent
```

The following example applies shading to the first cell in the first row.

```
ActiveDocument.Tables(1).Rows(1).Cells(1).Shading _
    .Texture = wdTexture20Percent
```

Use the [Add](#) method to add a **Cell** object to the **Cells** collection. You can also use the [InsertCells](#) method of the **Selection** object to insert new cells. The following example adds a cell before the first cell in myTable.

```
Set myTable = ActiveDocument.Tables(1)
myTable.Range.Cells.Add BeforeCell:=myTable.Cell(1, 1)
```

The following example sets a range (myRange) that references the first two cells in the first table. After the range is set, the cells are combined by the **Merge** method.

```
Set myTable = ActiveDocument.Tables(1)
Set myRange = ActiveDocument.Range(myTable.Cell(1, 1) _
    .Range.Start, myTable.Cell(1, 2).Range.End)
myRange.Cells.Merge
```

Remarks

Use the **Add** method with the **Rows** or **Columns** collection to add a row or column of cells.

Use the [Information](#) property with a **Selection** object to return the current row and column number. The following example changes the width of the first cell in the selection and then displays the cell's row number and column number.

```
If Selection.Information(wdWithInTable) = True Then
    With Selection
        .Cells(1).Width = 22
        MsgBox "Cell " & .Information(wdStartOfRangeRowNumber) _
            & ", " & .Information(wdStartOfRangeColumnNumber)
    End With
End If
```



Cells Collection Object

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

A collection of [Cell](#) objects in a table column, table row, selection, or range.

Using the Cells Object

Use the **Cells** property to return the **Cells** collection. The following example formats the cells in the first row in table one in the active document to be 30 points wide.

```
ActiveDocument.Tables(1).Rows(1).Cells.Width = 30
```

The following example returns the number of cells in the current row.

```
num = Selection.Rows(1).Cells.Count
```

Use the [Add](#) method to add a **Cell** object to the **Cells** collection. You can also use the [InsertCells](#) method of the **Selection** object to insert new cells. The following example adds a cell before the first cell in myTable.

```
Set myTable = ActiveDocument.Tables(1)
myTable.Range.Cells.Add BeforeCell:=myTable.Cell(1, 1)
```

Use **Cell**(*row*, *column*), where *row* is the row number and *column* is the column number, or **Cells**(*index*), where *index* is the index number, to return a **Cell** object. The following example applies shading to the second cell in the first row in table one.

```
Set myCell = ActiveDocument.Tables(1).Cell(Row:=1, Column:=2)
myCell.Shading.Texture = wdTexture20Percent
```

The following example applies shading to the first cell in the first row.

```
ActiveDocument.Tables(1).Rows(1).Cells(1).Shading _
    .Texture = wdTexture20Percent
```

Remarks

Use the **Add** method with the **Rows** or **Columns** collection to add a row or column of cells. The following example adds a column to the first table in the active document and then inserts numbers into the first column.

```
Set myTable = ActiveDocument.Tables(1)
Set aColumn = myTable.Columns.Add(BeforeColumn:=myTable.Columns(1))
For Each aCell In aColumn.Cells
    aCell.Range.Delete
    aCell.Range.InsertAfter num + 1
    num = num + 1
Next aCell
```



↳ [Show All](#)

Characters Collection Object

Multiple objects [↳ Characters \(Range\)](#)
↳ Multiple objects

A collection of characters in a selection, range, or document. There is no Character object; instead, each item in the **Characters** collection is a **Range** object that represents one character.

Using the Characters Collection

Use the **Characters** property to return the **Characters** collection. The following example displays how many characters are selected.

```
MsgBox Selection.Characters.Count & " characters are selected"
```

Use **Characters(index)**, where *index* is the index number, to return a **Range** object that represents one character. The index number represents the position of a character in the **Characters** collection. The following example formats the first letter in the selection as 24-point bold.

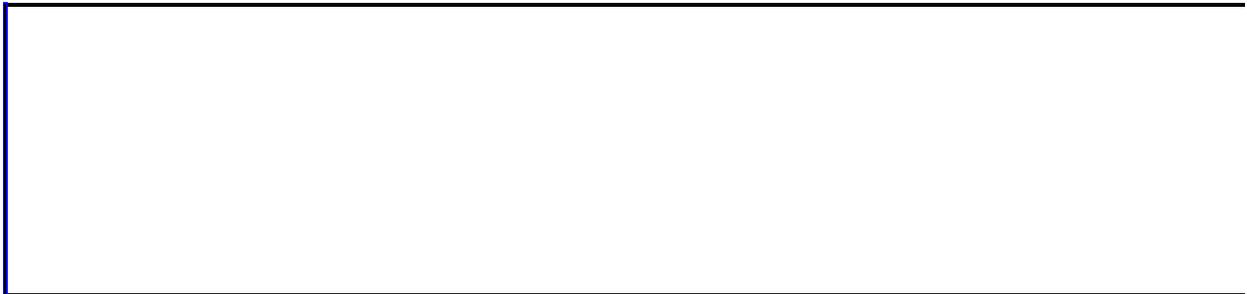
```
With Selection.Characters(1)  
    .Bold = True  
    .Font.Size = 24  
End With
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main [story](#) only. To count items in other stories use the collection with the **Range** object.

An **Add** method isn't available for the **Characters** collection. Instead, use the [InsertAfter](#) or [InsertBefore](#) method to add characters to a **Range** object. The following example inserts a new paragraph after the first paragraph in the active document.

```
With ActiveDocument
    .Paragraphs(1).Range.InsertParagraphAfter
    .Paragraphs(2).Range.InsertBefore "New Text"
End With
```



CheckBox Object

[FormFields \(FormField\)](#)  [CheckBox](#)

Represents a single check box form field.

Using the CheckBox Object

Use **FormFields**(*index*), where *index* is index number or the bookmark name associated with the check box, to return a single **FormField** object. Use the [CheckBox](#) property with the **FormField** object to return a **CheckBox** object. The following example selects the check box form field named "Check1" in the active document.

```
ActiveDocument.FormFields("Check1").CheckBox.Value = True
```

The index number represents the position of the form field in the [FormFields](#) collection. The following example checks the type of the first form field; if it's a check box, the check box is selected.

```
If ActiveDocument.FormFields(1).Type = wdFieldFormCheckBox Then  
    ActiveDocument.FormFields(1).CheckBox.Value = True  
End If
```

The following example determines whether the `ffield` object is valid before changing the check box size to 14 points.

```
Set ffield = ActiveDocument.FormFields(1).CheckBox  
If ffield.Valid = True Then  
    ffield.AutoSize = False  
    ffield.Size = 14  
Else  
    MsgBox "First field is not a check box"  
End If
```

Use the [Add](#) method with the **FormFields** object to add a check box form field. The following example adds a check box at the beginning of the active document, sets the name to "Color", and then selects the check box.

```
With ActiveDocument.FormFields.Add(Range:=ActiveDocument.Range _  
    (Start:=0,End:=0), Type:=wdFieldFormCheckBox)  
    .Name = "Color"  
    .CheckBox.Value = True  
End With
```



ColorFormat Object

Multiple objects `└ ColorFormat`

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.

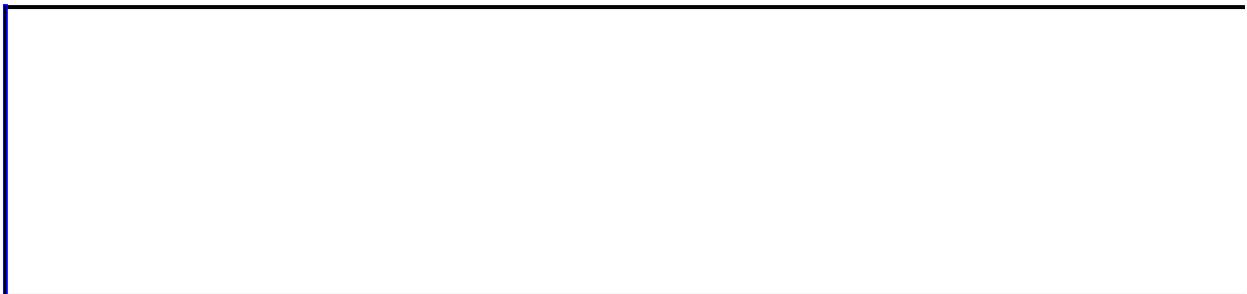
Using the ColorFormat Object

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 document and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
With ActiveDocument.Shapes _  
    .AddShape(msoShapeRectangle, 90, 90, 90, 50).Fill  
    .ForeColor.RGB = RGB(128, 0, 0)  
    .BackColor.RGB = RGB(170, 170, 170)  
    .TwoColorGradient msoGradientHorizontal, 1  
End With
```



Column Object

Multiple objects [└Columns \(Column\)](#)
└Multiple objects

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 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 column one in table one in the active document.

```
ActiveDocument.Tables(1).Columns(1).Select
```

Use the [Column](#) property with a **Cell** object to return a **Column** object. The following example deletes the text in cell one, inserts new text, and then sorts the entire column.

```
With ActiveDocument.Tables(1).Cell(1, 1)
    .Range.Delete
    .Range.InsertBefore "Sales"
    .Column.Sort
End With
```

Use the [Add](#) method to add a column to a table. The following example adds a column to the first table in the active document, and then it makes the column widths equal.

```
If ActiveDocument.Tables.Count >= 1 Then
    Set myTable = ActiveDocument.Tables(1)
    myTable.Columns.Add BeforeColumn:=myTable.Columns(1)
    myTable.Columns.DistributeWidth
End If
```

Remarks

Use the [Information](#) property with a **Selection** object to return the current column number. The following example selects the current column and then displays the column number in a message box.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Columns(1).Select
    MsgBox "Column " & Selection.Information(wdStartOfRangeColumnNumber)
End If
```



Columns Collection Object

Multiple objects [└Columns \(Column\)](#)
└Multiple objects

A collection of [Column](#) objects that represent the columns in a table.

Using the Columns Collection

Use the **Columns** property 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.

```
MsgBox ActiveDocument.Tables(1).Columns.Count
```

The following example creates a table with six columns and three rows and then formats each column with a progressively larger (darker) shading percentage.

```
Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _  
    NumRows:=3, NumColumns:=6)  
For Each col In myTable.Columns  
    col.Shading.Texture = 2 + i  
    i = i + 1  
Next col
```

Use the [Add](#) method to add a column to a table. The following example adds a column to the first table in the active document, and then it makes the column widths equal.

```
If ActiveDocument.Tables.Count >= 1 Then  
    Set myTable = ActiveDocument.Tables(1)  
    myTable.Columns.Add BeforeColumn:=myTable.Columns(1)  
    myTable.Columns.DistributeWidth  
End If
```

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 first column in the first table.

```
ActiveDocument.Tables(1).Columns(1).Select
```



Comment Object

Multiple objects [Comments \(Comment\)](#)
[Range](#)

Represents a single comment. The **Comment** object is a member of the [Comments](#) collection. The **Comments** collection includes comments in a selection, range or document.

Using the Comment Object

Use **Comments**(*index*), where *index* is the index number, to return a single **Comment** object. The index number represents the position of the comment in the specified selection, range, or document. The following example displays the author of the first comment in the active document.

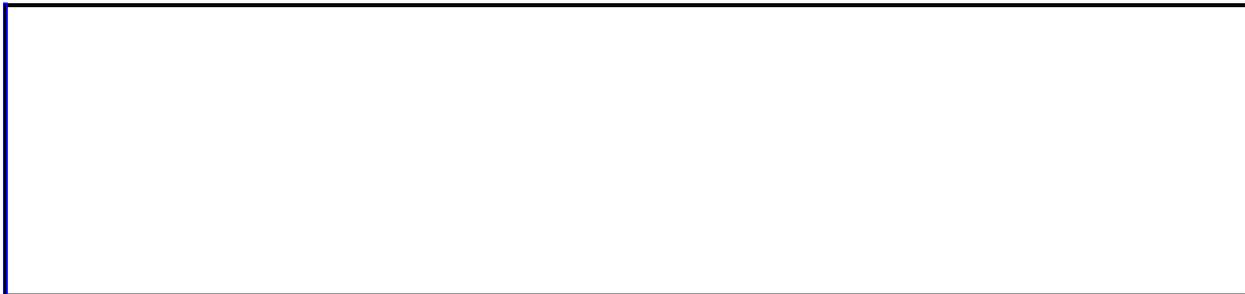
```
MsgBox ActiveDocument.Comments(1).Author
```

Use the **Add** method to add a comment at the specified range. The following example adds a comment immediately after the selection.

```
Selection.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Comments.Add Range:=Selection.Range, _  
    Text:="review this"
```

Use the **Reference** property to return the reference mark associated with the specified comment. Use the **Range** property to return the text associated with the specified comment. The following example displays the text associated with the first comment in the active document.

```
MsgBox ActiveDocument.Comments(1).Range.Text
```



Comments Collection Object

Multiple objects [↳ Comments \(Comment\)](#)
[↳ Range](#)

A collection of [Comment](#) objects that represent the comments in a selection, range, or document.

Using the Comments Collection

Use the **Comments** property to return the **Comments** collection. The following example displays comments made by Don Funk in the active document.

```
ActiveDocument.ActiveWindow.View.SplitSpecial = wdPaneComments
ActiveDocument.Comments.ShowBy = "Don Funk"
```

Use the [Add](#) method to add a comment at the specified range. The following example adds a comment immediately after the selection.

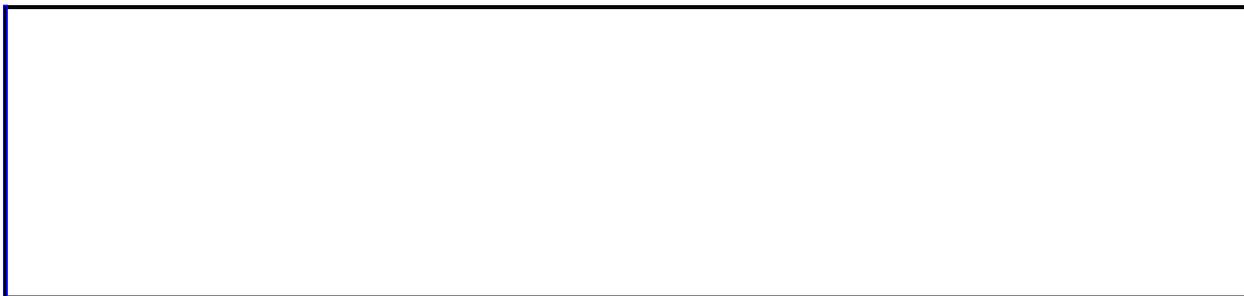
```
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.Comments.Add Range:=Selection.Range, _
    Text:="review this"
```

Use **Comments(index)**, where *index* is the index number, to return a single **Comment** object. The index number represents the position of the comment in the specified selection, range, or document. The following example displays the author of the first comment in the active document.

```
MsgBox ActiveDocument.Comments(1).Author
```

The following example displays the initials of the author of the first comment in the selection.

```
If Selection.Comments.Count >= 1 Then MsgBox _
    Selection.Comments(1).Initial
```



ConditionalStyle Object

[TableStyle](#) └ [ConditionalStyle](#)
└ Multiple objects

Represents special formatting applied to specified areas of a table when the selected table is formatted with a specified table style.

Using the ConditionalStyle object

Use the [Condition](#) method of the [TableStyle](#) object to return a **ConditionalStyle** object. The [Shading](#) property can be used to apply shading to specified areas of a table. This example selects the first table in the active document and applies shading to alternate rows and columns. This example assumes that there is a table in the active document and that it is formatted using the Table Grid style.

```
Sub ApplyConditionalStyle()  
  With ActiveDocument  
    .Tables(1).Select  
    With .Styles("Table Grid").Table  
      .Condition(wdOddColumnBanding).Shading _  
        .BackgroundColor = wdColorGray10  
      .Condition(wdOddRowBanding).Shading _  
        .BackgroundColor = wdColorGray10  
    End With  
  End With  
End Sub
```

Use the [Borders](#) property to apply borders to specified areas of a table. This example selects the first table in the active document and applies borders to the first and last row and first column. This example assumes that there is a table in the active document and that it is formatted using the Table Grid style.

```
Sub ApplyTableBorders()  
  With ActiveDocument  
    .Tables(1).Select  
    With .Styles("Table Grid").Table  
      .Condition(wdFirstRow).Borders(wdBorderBottom) _  
        .LineStyle = wdLineStyleDouble  
      .Condition(wdFirstColumn).Borders(wdBorderRight) _  
        .LineStyle = wdLineStyleDouble  
      .Condition(wdLastRow).Borders(wdBorderTop) _  
        .LineStyle = wdLineStyleDouble  
    End With  
  End With  
End Sub
```



CustomLabel Object

[Application](#) └ [MailingLabel](#)
└ [CustomLabels \(CustomLabel\)](#)

Represents a custom mailing label. The **CustomLabel** object is a member of the **CustomLabels** collection. The **CustomLabels** collection contains all the custom mailing labels listed in the **Label Options** dialog box.

Using the CustomLabel Object

Use **CustomLabels**(*index*), where *index* is the custom label name or index number, to return a single **CustomLabel** object. The following example creates a new document with an existing custom label layout named "My Labels."

```
Set ML = Application.MailingLabel
If ML.CustomLabels("My Labels").Valid = True Then
    ML.CreateNewDocument Name:="My Labels"
Else
    MsgBox "The My Labels custom label is not available"
End If
```

The index number represents the position of the custom mailing label in the **CustomLabels** collection. The following example displays the name of the first custom mailing label.

```
If Application.MailingLabel.CustomLabels.Count >= 1 Then
    MsgBox Application.MailingLabel.CustomLabels(1).Name
End If
```

Note **CustomLabel** objects are sorted alphabetically in the **CustomLabels** collection and their index numbers are dynamically reassigned as the contents of the collection change. For that reason, it is safer to refer to a specific **CustomLabel** object by name rather than by index number.

Use the [Add](#) method to create a custom label. The following example adds a custom mailing label named "My Label" and sets the page size.

```
Set ML = _
    Application.MailingLabel.CustomLabels.Add(Name:="My Labels", _
        DotMatrix:=False)
ML.PageSize = wdCustomLabelA4
```



CustomLabels Collection Object

[Application](#) └ [MailingLabel](#)
└ [CustomLabels \(CustomLabel\)](#)

A collection of [CustomLabel](#) objects available in the **Label Options** dialog box. This collection includes custom labels of all printer types (dot-matrix, laser, and ink-jet printers).

Using the CustomLabels Collection

Use the **CustomLabels** property to return the **CustomLabels** collection. The following example displays the number of available custom labels.

```
MsgBox Application.MailingLabel.CustomLabels.Count
```

Use the [Add](#) method to create a custom label. The following example adds a custom mailing label named "My Label" and sets the page size.

```
Set ML = _  
    Application.MailingLabel.CustomLabels.Add(Name:="My Labels", _  
        DotMatrix:=False)  
ML.PageSize = wdCustomLabelA4
```

Use **CustomLabels(index)**, where *index* is the custom label name or index number, to return a single **CustomLabel** object. The following example creates a new document with an existing custom label layout named "My Labels."

```
Set ML = Application.MailingLabel  
If ML.CustomLabels("My Labels").Valid = True Then  
    ML.CreateNewDocument Name:="My Labels"  
Else  
    MsgBox "The My Labels custom label is not available"  
End If
```

The index number represents the position of the custom mailing label in the **CustomLabels** collection. The following example displays the name of the first custom mailing label.

```
If Application.MailingLabel.CustomLabels.Count >= 1 Then  
    MsgBox Application.MailingLabel.CustomLabels(1).Name  
End If
```



CustomProperties Collection

[SmartTag](#) [└ CustomProperties](#)
[└ CustomProperty](#)

A collection of [CustomProperty](#) objects that represents the properties related to a smart tag. The **CustomProperties** collection includes all the smart tag custom properties in a document.

Using the CustomProperties collection

Use the [Properties](#) property to return a single **CustomProperties** object. Use the [Add](#) method of the **CustomProperties** object with to create a custom property from within a Microsoft Word Visual Basic for Applications project. This example creates a new property for the first smart tag in the active document and displays the XML code used for the tag.

```
Sub AddProps()  
    With ThisDocument.SmartTags(1)  
        .Properties.Add Name:="President", Value:=True  
        MsgBox "The XML code is " & .XML  
    End With  
End Sub
```

Use [Properties\(index\)](#) to return a single property for a smart tag, where *index* is the number of the property. This example displays the name and value of the first property of the first smart tag in the current document.

```
Sub ReturnProps()  
    With ThisDocument.SmartTags(1).Properties(1)  
        MsgBox "The Smart Tag name is: " & .Name & vbLf & .Value  
    End With  
End Sub
```

Use the [Count](#) property to return the number of custom properties for a smart tag. This example loops through all the smart tags in the current document and then lists in a new document the name and value of the custom properties for all smart tags that have custom properties.

```
Sub SmartTagsProps()  
    Dim docNew As Document  
    Dim stgTag As SmartTag  
    Dim stgProp As CustomProperty  
    Dim intTag As Integer  
    Dim intProp As Integer  
  
    Set docNew = Documents.Add  
  
    'Create heading info in new document  
    With docNew.Content  
        .InsertAfter "Name" & vbTab & "Value"
```

```

        .InsertParagraphAfter
    End With

    'Loop through smart tags in current document
    For intTag = 1 To ThisDocument.SmartTags.Count

        With ThisDocument.SmartTags(intTag)

            'Verify that the custom properties
            'for smart tags is greater than zero
            If .Properties.Count > 0 Then

                'Loop through the custom properties
                For intProp = 1 To .Properties.Count

                    'Add custom property name to new document
                    docNew.Content.InsertAfter .Properties(intProp)
                        .Name & vbTab & .Properties(intProp).Value
                    docNew.Content.InsertParagraphAfter

                Next

            Else

                'Display message if there are no custom properties
                MsgBox "There are no custom properties for the " & _
                    "smart tags in your document."

            End If

        End With

    Next

    'Convert the content in the new document into a table
    docNew.Content.Select
    Selection.ConvertToTable Separator:=wdSeparateByTabs, NumColumns

End Sub

```



CustomProperty Object

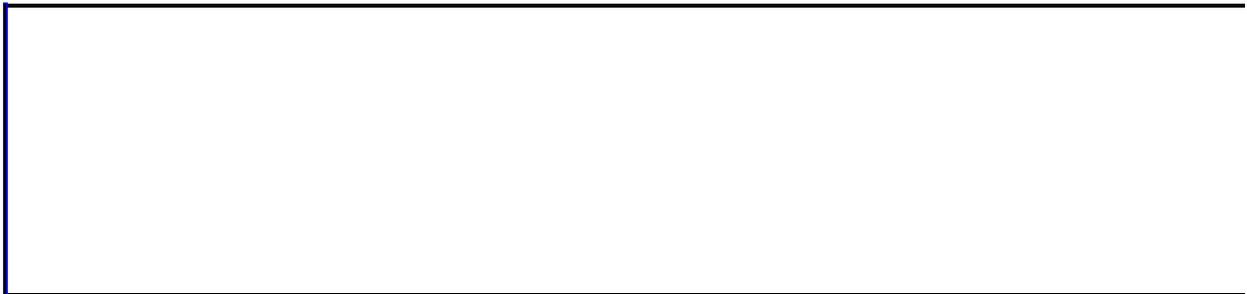
[CustomProperties](#) | [CustomProperty](#)

Represents a single instance of a custom property for a smart tag. The **CustomProperty** object is a member of the [CustomProperties](#) collection.

Using the CustomProperty object

Use the [Item](#) method — or [Properties\(Index\)](#), where *index* is the number of the property — of the **CustomProperties** collection to return a **CustomProperty** object. Use the [Name](#) and [Value](#) properties to return the information related to a custom property for a smart tag. This example displays a message containing the name and value of the first custom property of the first smart tag in the current document. This example assumes that the current document contains at least one smart tag and that the first smart tag has at least one custom property.

```
Sub SmartTagsProps()  
    With ThisDocument.SmartTags(Index:=1).Properties.Item(Index:=1)  
        MsgBox "Smart Tag Name: " & .Name & vbLf & _  
            "Smart Tag Value: " & .Value  
    End With  
End Sub
```



DefaultWebOptions Object

[Application](#) └ [DefaultWebOptions](#)

Contains global application-level attributes used by Microsoft Word when you save a document as a Web page or open a Web page. You can return or set attributes either at the application (global) level or at the document level. (Note that attribute values can be different from one document to another, depending on the attribute value at the time the document was saved.) Document-level attribute settings override application-level attribute settings. Document-level attributes are contained in the [WebOptions](#) object.

Using the DefaultWebOptions Object

Use the [DefaultWebOptions](#) method to return the **DefaultWebOptions** object. The following example checks to see whether PNG (Portable Network Graphics) is allowed as an image format and sets the strImageFileType variable accordingly.

```
Set objAppWebOptions = Application.DefaultWebOptions
With objAppWebOptions
    If .AllowPNG = True Then
        strImageFileType = "PNG"
    Else
        strImageFileType = "JPG"
    End If
End With
```



Diagram Object

[DiagramNode](#) └ [Diagram](#)
└ [DiagramNodes](#)

Represents a single diagram in a document. The **Diagram** object is a member of the [Shapes](#) collection.

Using the Diagram object

Use the [Diagram](#) property of the [DiagramNode](#), [Shape](#), and [ShapeRange](#) objects to return a single **Diagram** object. Use the [Convert](#) method to change a diagram from one type to another. This example converts the first diagram in the active document into a radial diagram. This example assumes that the first shape in the active document is a diagram and not another type of shape.

```
Sub DiagramConvert()  
    ActiveDocument.Shapes(1).Diagram.Convert msoDiagramRadial  
End Sub
```

Use the [Reverse](#) property to flip the order of the nodes in a diagram. This example reverses the order of the diagram nodes in the second shape in the active document. This assumes that the second shape in the active document is a diagram.

```
Sub DiagramReverse()  
    ActiveDocument.Shapes(2).Diagram.Reverse = msoTrue  
End Sub
```



DiagramNode Object

Multiple objects [↳DiagramNode](#)
↳Multiple objects

Represents a single diagram node within a diagram. The **DiagramNode** object is a member of the [DiagramNodes](#) collection.

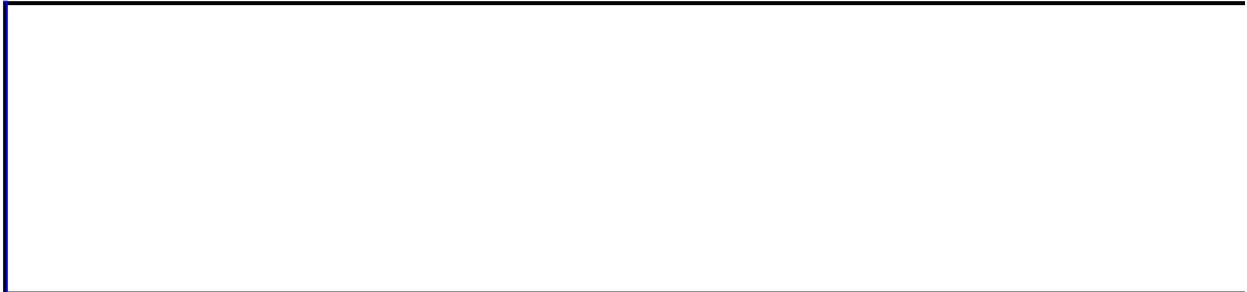
Using the DiagramNode object

Use the [DiagramNode](#) property of the [Shape](#) or [ShapeRange](#) object to return a **DiagramNode** object. Use the [AddNode](#) method to add a node to a diagram. This example assumes the third shape in the document is a diagram and adds a node to it.

```
Sub AddDiagramNode()  
    ActiveDocument.Shapes(3).DiagramNode.Children.AddNode  
End Sub
```

Use the [Delete](#) method to remove a node from a diagram. This example assumes the second shape in the document is a diagram and removes the first node from it.

```
Sub DeleteDiagramNode()  
    ActiveDocument.Shapes(2).DiagramNode.Children(1).Delete  
End Sub
```



DiagramNodeChildren Collection

[DiagramNode](#) └ [DiagramNodeChildren](#)
└ [DiagramNode](#)

A collection of **DiagramNode** objects that represents the child nodes in a diagram.

Using the DiagramNodeChildren collection

Use the [Children](#) property to return the nodes in a **DiagramNodeChildren** collection. Use the [FirstChild](#) property to access the first child node in a diagram. This example deletes the first child of the second node in the first diagram in the document. This example assumes that the first shape in the active document is a diagram with at least two nodes, one with child nodes.

```
Sub DiagramNodeChild()  
    ActiveDocument.Shapes(1).Diagram.Nodes.Item(2) _  
        .Children.FirstChild.Delete  
End Sub
```



DiagramNodes Collection

[Diagram](#) └ [DiagramNodes](#)
└ [DiagramNode](#)

A collection of [DiagramNode](#) objects that represent all the nodes in a diagram. The **DiagramNodes** collection contains all the diagram nodes in a specified diagram.

Using the DiagramNodes collection

Use the [Nodes](#) property to return the **DiagramNodes** collection. Use the [SelectAll](#) method to select and work with all nodes in a diagram. This example selects all nodes in the specified diagram and fills them with the specified pattern. The following example assumes the first shape in the active document is a diagram.

```
Sub FillDiagramNodes()  
    ActiveDocument.Shapes(1).Diagram.Nodes.SelectAll  
    Selection.ShapeRange.Fill.Patterned msoPatternSmallConfetti  
End Sub
```

Use the [Item](#) method to select and work with a single diagram node in a diagram. This example selects the first node in the specified diagram and deletes it. The following example assumes the first shape in the active document is a diagram.

```
Sub FillDiagramNode()  
    ActiveDocument.Shapes(1).Diagram.Nodes.Item(1).Delete  
End Sub
```



Dialog Object

[Application](#) | [Dialogs \(Dialog\)](#)

Represents a built-in dialog box. The **Dialog** object is a member of the **Dialogs** collection. The **Dialogs** collection contains all the built-in dialog boxes in Word. You cannot create a new built-in dialog box or add one to the **Dialogs** collection.

Using the Dialog Object

Use **Dialogs**(*index*), where *index* is a **WdWordDialog** constant that identifies the dialog box, to return a single **Dialog** object. The following example displays and carries out the actions taken in the built-in **Open** dialog box (**File** menu).

```
dlgAnswer = Dialogs(wdDialogFileOpen).Show
```

The **WdWordDialog** constants are formed from the prefix "wdDialog" followed by the name of the menu and the dialog box. For example, the constant for the **Page Setup** dialog box is **wdDialogFilePageSetup**, and the constant for the **New** dialog box is **wdDialogFileNew**. For more information about working with built-in Word dialog boxes, see [Displaying built-in Word dialog boxes](#).



Dialogs Collection Object

[Application](#) | [Dialogs \(Dialog\)](#)

A collection of **Dialog** objects in Word. Each **Dialog** object represents a built-in Word dialog box.

Using the Dialogs Collection

Use the **Dialogs** property to return the **Dialogs** collection. The following example displays the number of available built-in dialog boxes.

```
MsgBox Dialogs.Count
```

You cannot create a new built-in dialog box or add one to the **Dialogs** collection. Use **Dialogs(index)**, where *index* is the **WdWordDialog** constant that identifies the dialog box, to return a single **Dialog** object. The following example displays the built-in **Open** dialog box.

```
dlgAnswer = Dialogs(wdDialogFileOpen).Show
```

For more information, see [Displaying built-in Word dialog boxes](#).



Dictionaries Collection Object

Multiple objects [↳ Dictionaries \(Dictionary\)](#)

A collection of [Dictionary](#) objects that includes the active custom spelling dictionaries.

Using the Dictionaries Collection

Use the [CustomDictionaries](#) property to return the collection of currently active custom dictionaries. The following example displays the names of all the active custom dictionaries.

```
For Each d In CustomDictionaries
    MsgBox d.Name
Next d
```

Use the [Add](#) method to add a new custom dictionary to the collection of active custom dictionaries. If there isn't a file with the name specified by *FileName*, Word creates it. The following example adds "MyCustom.dic" to the collection of custom dictionaries.

```
CustomDictionaries.Add FileName:="MyCustom.dic"
```

Use the [ClearAll](#) method to unload all custom dictionaries. Note, however, that this method doesn't delete the dictionary files. After you use this method, the number of custom dictionaries in the collection is 0 (zero). The following example clears the custom dictionaries and creates a new custom dictionary file. The new dictionary is set as the active custom dictionary, to which Word will automatically add any new words it encounters.

```
With CustomDictionaries
    .ClearAll
    .Add FileName:= "MyCustom.dic"
    .ActiveCustomDictionary = CustomDictionaries(1)
End With
```

Remarks

You set the custom dictionary to which new words are added by using the [ActiveCustomDictionary](#) property. If you try to set this property to a dictionary that isn't a custom dictionary, an error occurs.

The [Maximum](#) property returns the maximum number of simultaneous custom spelling dictionaries that the application can support. For Word, this maximum is 10.



Dictionary Object

Multiple objects [↳ Dictionaries \(Dictionary\)](#)

Represents a dictionary. **Dictionary** objects that represent custom dictionaries are members of the [Dictionaries](#) collection. Other dictionary objects are returned by properties of the **Languages** collection; these include the **ActiveSpellingDictionary**, **ActiveGrammarDictionary**, **ActiveThesaurusDictionary**, and **ActiveHyphenationDictionary** properties.

Using the Dictionary Object

Use **CustomDictionaries**(*index*), where *index* is an index number or the string name for the dictionary, to return a single **Dictionary** object that represents a custom dictionary. The following example returns the first dictionary in the collection.

```
CustomDictionaries(1)
```

The following example returns the dictionary named "MyDictionary."

```
CustomDictionaries("MyDictionary")
```

Use the [ActiveCustomDictionary](#) property to set the custom spelling dictionary in the collection to which new words are added. If you try to set this property to a dictionary that's not a custom dictionary, an error occurs.

Use the [Add](#) method to add a new dictionary to the collection of active custom dictionaries. If there's no file with the name specified by **FileName**, Word creates it. The following example adds "MyCustom.dic" to the collection of custom dictionaries.

```
CustomDictionaries.Add FileName:="MyCustom.dic"
```

Remarks

Use the [Name](#) and [Path](#) properties to locate any of the dictionaries. The following example displays a message box that contains the full path for each dictionary.

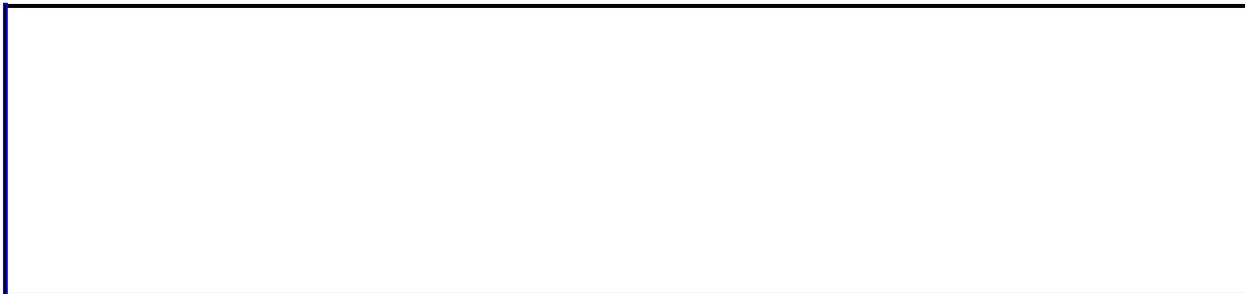
```
For Each d in CustomDictionaries
    MsgBox d.Path & Application.PathSeparator & d.Name
Next d
```

Use the [LanguageSpecific](#) property to determine whether the specified custom dictionary can have a specific language assigned to it with the [LanguageID](#) property. If the dictionary is language specific, it will verify only text that's formatted for the specified language.

For each language for which proofing tools are installed, you can use the [ActiveGrammarDictionary](#), [ActiveHyphenationDictionary](#), [ActiveSpellingDictionary](#), and [ActiveThesaurusDictionary](#) properties to return the corresponding **Dictionary** objects. The following example returns the full path for the active spelling dictionary used in the U.S. English version of Word.

```
Set myspell = Languages(wdEnglishUS).ActiveSpellingDictionary
MsgBox mySpell.Path & Application.PathSeparator & mySpell.Name
```

The [ReadOnly](#) property returns **True** for .lex files (built-in proofing dictionaries) and **False** for .dic files (custom spelling dictionaries).



Document Object

Multiple objects [└ Documents \(Document\)](#)
└ Multiple objects

Represents a document. The **Document** object is a member of the [Documents](#) collection. The **Documents** collection contains all the **Document** objects that are currently open in Word.

Using the Document Object

Use **Documents**(*index*), where *index* is the document name or index number to return a single **Document** object. The following example closes the document named "Report.doc" without saving changes.

```
Documents("Report.doc").Close SaveChanges:=wdDoNotSaveChanges
```

The index number represents the position of the document in the **Documents** collection. The following example activates the first document in the **Documents** collection.

```
Documents(1).Activate
```

Using ActiveDocument

You can use the **ActiveDocument** property to refer to the document with the focus. The following example uses the **Activate** method to activate the document named "Document 1." The example also sets the page orientation to landscape mode and then prints the document.

```
Documents("Document1").Activate  
ActiveDocument.PageSetup.Orientation = wdOrientLandscape  
ActiveDocument.PrintOut
```



Documents Collection Object

[Application](#) └ [Documents \(Document\)](#)
└ Multiple objects

A collection of all the [Document](#) objects that are currently open in Word.

Using the Documents Collection

Use the **Documents** property to return the **Documents** collection. The following example displays the names of the open documents.

```
For Each aDoc In Documents
    aName = aName & aDoc.Name & vbCr
Next aDoc
MsgBox aName
```

Use the [Add](#) method to create a new empty document and add it to the **Documents** collection. The following example creates a new document based on the Normal template.

```
Documents.Add
```

Use the **Open** method to open a file. The following example opens the document named "Sales.doc."

```
Documents.Open FileName:="C:\My Documents\Sales.doc"
```

Use **Documents(index)**, where *index* is the document name or index number to return a single **Document** object. The following instruction closes the document named "Report.doc" without saving changes.

```
Documents("Report.doc").Close SaveChanges:=wdDoNotSaveChanges
```

The index number represents the position of the document in the **Documents** collection. The following example activates the first document in the **Documents** collection.

```
Documents(1).Activate
```

Remarks

The following example enumerates the **Documents** collection to determine whether the document named "Report.doc" is open. If this document is contained in the **Documents** collection, the document is activated; otherwise, it's opened.

```
For Each doc In Documents
    If doc.Name = "Report.doc" Then found = True
Next doc
If found <> True Then
    Documents.Open FileName:="C:\Documents\Report.doc"
Else
    Documents("Report.doc").Activate
End If
```



DropCap Object

Multiple objects [└ Paragraphs \(Paragraph\)](#)
[└ DropCap](#)

Represents a dropped capital letter at the beginning of a paragraph. There is no DropCaps collection; each **Paragraph** object contains only one **DropCap** object.

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 in the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).DropCap  
    .Enable  
    .Position = wdDropNormal  
End With
```



DropDown Object

[Documents \(Document\)](#) [FormFields \(FormField\)](#)
└ [DropDown](#)
 └ [ListEntries \(ListEntry\)](#)

Represents a drop-down form field that contains a list of items in a form.

Using the DropDown Object

Use **FormFields**(*index*), where *index* is the index number or the bookmark name associated with the drop-down form field, to return a single **FormField** object. Use the **DropDown** property with the **FormField** object to return a **DropDown** object. The following example selects the first item in the drop-down form field named "DropDown" in the active document.

```
ActiveDocument.FormFields("DropDown1").DropDown.Value = 1
```

The index number represents the position of the form field in the **FormFields** collection. The following example checks the type of the first form field in the active document. If it's a drop-down form field, the second item is selected.

```
If ActiveDocument.FormFields(1).Type = wdFieldFormDropDown Then
    ActiveDocument.FormFields(1).DropDown.Value = 2
End If
```

The following example determines whether form field represented by *ffield* is a valid drop-down form field before adding an item to it.

```
Set ffield = ActiveDocument.FormFields(1).DropDown
If ffield.Valid = True Then
    ffield.ListEntries.Add Name:="Hello"
Else
    MsgBox "First field is not a drop down"
End If
```

Use the **Add** method with the **FormFields** collection to add a drop-down form field. The following example adds a drop-down form field at the beginning of the active document and then adds items to the form field.

```
Set ffield = ActiveDocument.FormFields.Add( _
    Range:=ActiveDocument.Range(Start:=0, End:=0), _
    Type:=wdFieldFormDropDown)
With ffield
    .Name = "Colors"
    With .DropDown.ListEntries
        .Add Name:="Blue"
        .Add Name:="Green"
        .Add Name:="Red"
    End With
End With
```

End With

--

Email Object

[Documents \(Document\)](#) [└Email](#)
[└EmailAuthor](#)

Represents an e-mail message. There is no Emails collection; each **Document** object contains only one **Email** object.

Using the Email Object

Use the [Email](#) property to return the **Email** object. The **Email** object and its properties are valid only if the active document is an unsent forward, reply, or new e-mail message.

This example returns the name of the style associated with the current e-mail author.

```
MsgBox ActiveDocument.Email _  
    .CurrentEmailAuthor.Style.NameLocal
```

Note The author style name is the same as the value returned by the [UserName](#) property.



EmailAuthor Object

[Email](#) └ [EmailAuthor](#)
└ [Style](#)

Represents the author of an e-mail message. There is no EmailAuthors collection; each [Email](#) object contains only one **EmailAuthor** object.

Using the EmailAuthor Object

Use the [CurrentEmailAuthor](#) property to return the **EmailAuthor** object. The **EmailAuthor** object and its properties are valid only if the active document is an unsent forward, reply, or new e-mail message.

This example returns the style associated with the current author for unsent replies, forwards, or new e-mail messages, and displays the name of the font associated with this style.

```
Set MyEmailStyle = _  
    ActiveDocument.Email.CurrentEmailAuthor.Style  
Msgbox MyEmailStyle.Font.Name
```



EmailOptions Object

[Application](#) └ [EmailOptions](#)
└ Multiple objects

Contains global application-level attributes used by Microsoft Word when you create and edit e-mail messages and replies.

Using the EmailOptions Object

Use the [EmailOptions](#) property to return the **EmailOptions** object.

This example changes the font color of the default style used to compose new e-mail messages.

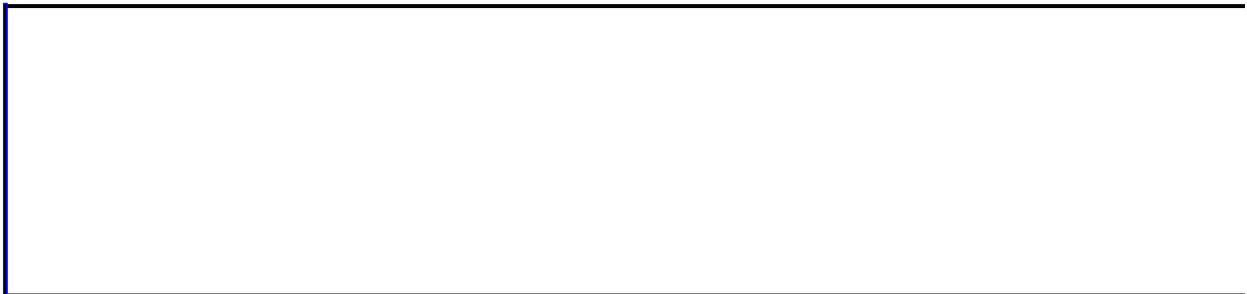
```
Application.EmailOptions.ComposeStyle.Font.Color = _  
    wdColorBrightGreen
```

This example sets Word to mark comments in e-mail messages with the initials "WK."

```
Application.EmailOptions.MarkCommentsWith = "WK"  
Application.EmailOptions.MarkComments = True
```

This example changes the signatures Word appends to new outgoing e-mail messages and e-mail message replies.

```
With Application.EmailOptions.EmailSignature  
    .NewMessageSignature = "Signature1"  
    .ReplyMessageSignature = "Reply2"  
End With
```



EmailSignature Object

[EmailOptions](#)  [EmailSignature](#)
 [EmailSignatureEntries](#)

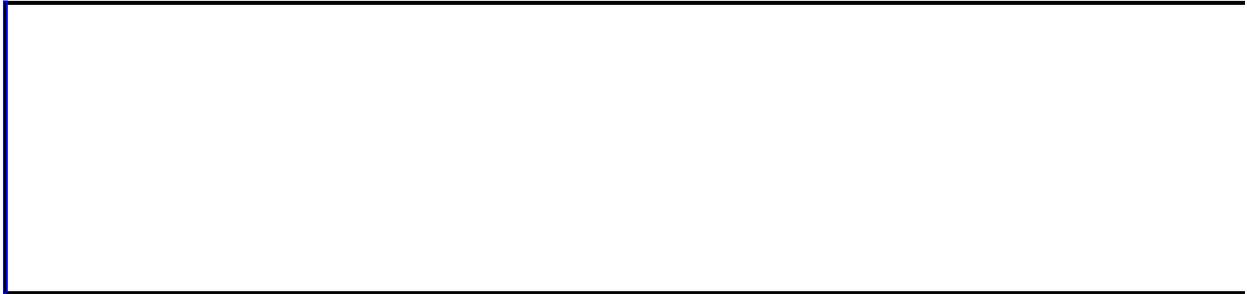
Contains information about the e-mail signatures used by Microsoft Word when you create and edit e-mail messages and replies. There is no `EmailSignatures` collection; each [EmailOptions](#) object contains only one **EmailSignature** object.

Using the EmailSignature Object

Use the [EmailSignature](#) property to return the **EmailSignature** object.

This example changes the signatures Word appends to new outgoing e-mail messages and e-mail message replies.

```
With Application.EmailOptions.EmailSignature  
    .NewMessageSignature = "Signature1"  
    .ReplyMessageSignature = "Reply2"  
End With
```



EmailSignatureEntries Collection

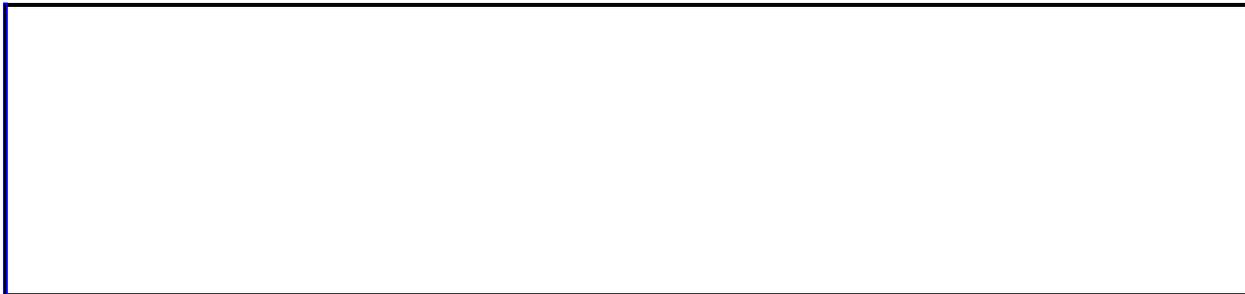
[EmailSignature](#)  [EmailSignatureEntries](#)
 [EmailSignatureEntry](#)

A collection of [EmailSignatureEntry](#) objects that represents all the e-mail signature entries available to Word.

Using the **EmailSignatureEntries** collection

Use the [EmailSignatureEntries](#) property to return the **EmailSignatureEntries** collection. Use the [Add](#) method of the **EmailSignatureEntries** object to add an e-mail signature to Word. The following example creates a new e-mail signature entry based on the author's name and a selection in the active document, and then it sets the new signature entry as the default e-mail signature to use for new messages.

```
Sub NewEmailSignature()  
    With Application.EmailOptions.EmailSignature  
        .EmailSignatureEntries.Add "Jeff Smith", Selection.Range  
        .NewMessageSignature = "Jeff Smith"  
    End With  
End Sub
```



EmailSignatureEntry Object

[EmailSignatureEntries](#)  [EmailSignatureEntry](#)

Represents a single e-mail signature entry. The **EmailSignatureEntry** object is a member of the [EmailSignatureEntries](#) collection. The **EmailSignatureEntries** collection contains all the e-mail signature entries available to Word.

Using the EmailSignatureEntry object

Use **EmailSignatureEntries**(*index*), where *index* is the e-mail signature entry name or item number, to return a single **EmailSignatureEntry** object. You must match exactly the spelling (but not necessarily the capitalization) of the name. The following example uses the [Delete](#) method to delete the signature entry named "Jeff Smith."

```
Sub DeleteSignature()  
    Application.EmailOptions.EmailSignature _  
        .EmailSignatureEntries("jeff smith").Delete  
End Sub
```



Endnote Object

Multiple objects [└ Endnotes \(Endnote\)](#)
[└ Range](#)

Represents an endnote. The **Endnote** object is a member of the [Endnotes](#) collection. The **Endnotes** collection represents the endnotes in a selection, range, or document.

Using the Endnote Object

Use **Endnotes**(*index*), where *index* is the index number, to return a single **Endnote** object. The index number represents the position of the endnote in the selection, range, or document. The following example applies red formatting to the first endnote in the selection.

```
If Selection.Endnotes.Count >= 1 Then
    Selection.Endnotes(1).Reference.Font.ColorIndex = wdRed
End If
```

Use the [Add](#) method to add an endnote to the **Endnotes** collection. The following example adds an endnote immediately after the selection.

```
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.Endnotes.Add Range:=Selection.Range , _
    Text:="The Willow Tree, (Lone Creek Press, 1996)."
```



EndnoteOptions Object

Multiple objects [└ EndnoteOptions](#)

Represents the properties assigned to a range or selection of endnotes in a document.

Using the EndnoteOptions object

Use the [Range](#) or [Selection](#) object to return an **EndnoteOptions** object. Using the **EndnoteOptions** object, you can assign different endnote properties to different areas of a document. For example, you may want endnotes in the introduction of a long document to be displayed as lowercase Roman numerals, while in the rest of your document they are displayed as Arabic numerals. The following example uses the [NumberingRule](#), [NumberStyle](#), and [StartingNumber](#) properties to format the endnotes in the first section of the active document.

```
Sub BookIntro()  
    Dim rngIntro As Range  
  
    'Sets the range as section one of the active document  
    Set rngIntro = ActiveDocument.Sections(1).Range  
  
    'Formats the EndnoteOptions properties  
    With rngIntro.EndnoteOptions  
        .NumberingRule = wdRestartSection  
        .NumberStyle = wdNoteNumberStyleLowercaseRoman  
        .StartingNumber = 1  
    End With  
End Sub
```



Endnotes Collection Object

Multiple objects [└ Endnotes \(Endnote\)](#)
[└ Range](#)

A collection of [Endnote](#) objects that represents all the endnotes in a selection, range, or document.

Using the Endnotes Collection

Use the **Endnotes** property to return the **Endnotes** collection. The following example sets the location of endnotes in the active document.

```
ActiveDocument.Endnotes.Location = wdEndOfSection
```

Use the [Add](#) method to add an endnote to the **Endnotes** collection. The following example adds an endnote immediately after the selection.

```
Selection.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Endnotes.Add Range:=Selection.Range , _  
    Text:="The Willow Tree, (Lone Creek Press, 1996)."
```

Use **Endnotes(index)**, where *index* is the index number, to return a single **Endnote** object. The index number represents the position of the endnote in a selection, range, or document. The following example applies red formatting to the first endnote in the selection.

```
If Selection.Endnotes.Count >= 1 Then  
    Selection.Endnotes(1).Reference.Font.ColorIndex = wdRed  
End If
```



Envelope Object

[Documents \(Document\)](#) [Envelope](#)
└ Multiple objects

Represents an envelope. There is no Envelopes collection; each **Document** object contains only one **Envelope** object.

Using the Envelope Object

Use the **Envelope** property to return the **Envelope** object. The following example adds an envelope to a new document and sets the distance between the top of the envelope and the address to 2.25 inches.

```
Set myDoc = Documents.Add
addr = "Michael Matey" & vbCr & "123 Skye St." _
      & vbCr & "Redmond, WA 98107"
retaddr = "Cora Edmonds" & vbCr & "456 Erde Lane" & vbCr _
        & "Redmond, WA 98107"
With myDoc.Envelope
    .Insert Address:=addr, ReturnAddress:=retaddr
    .AddressFromTop = InchesToPoints(2.25)
End With
```

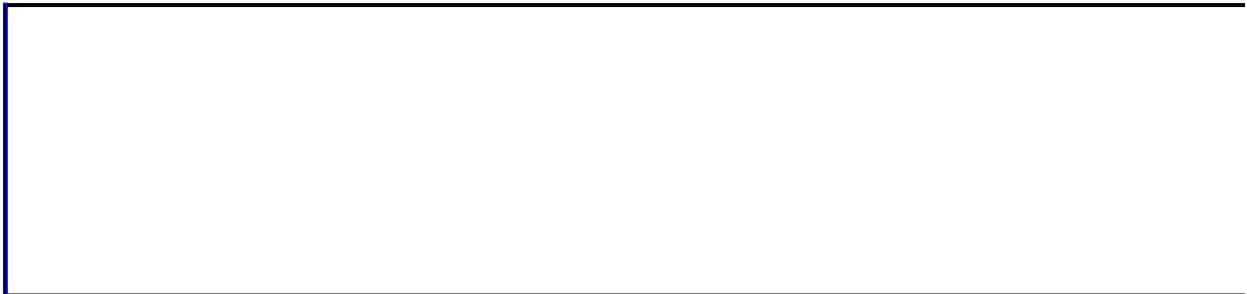
Remarks

The **Envelope** object is available regardless of whether an envelope has been added to the specified document. However, an error occurs if you use one of the following properties when an envelope hasn't been added to the document: **Address**, **AddressFromLeft**, **AddressFromTop**, **FeedSource**, **ReturnAddress**, **ReturnAddressFromLeft**, **ReturnAddressFromTop**, and **UpdateDocument**.

The following example demonstrates how to use the **On Error GoTo** statement to trap the error that occurs if an envelope hasn't been added to the active document. If, however, an envelope has been added to the document, the recipient address is displayed.

```
On Error GoTo ErrorHandler
MsgBox ActiveDocument.Envelope.Address
ErrorHandler:
If Err = 5852 Then MsgBox _
    "Envelope is not in the specified document"
```

Use the [Insert](#) method to add an envelope to the specified document. Use the [PrintOut](#) method to set the properties of an envelope and print it without adding it to the document.



Field Object

Multiple objects [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 document.

Using the Field Object

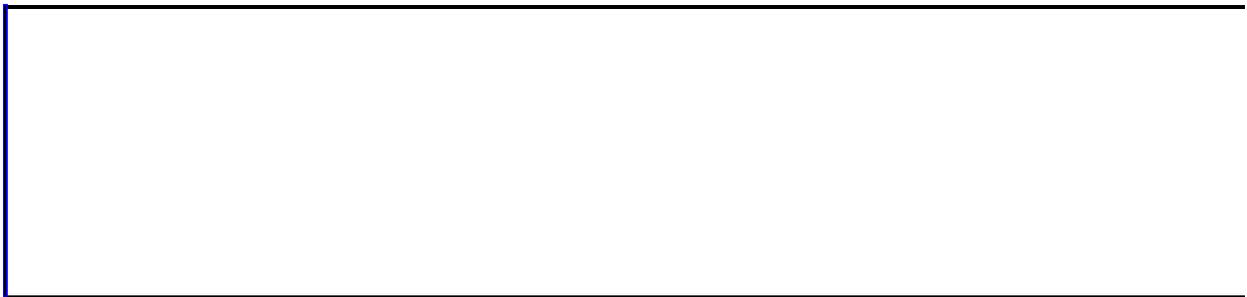
Use **Fields**(*index*), where *index* is the index number, to return a single **Field** object. The index number represents the position of the field in the selection, range, or document. The following example displays the field code and the result of the first field in the active document.

```
If ActiveDocument.Fields.Count >= 1 Then
    MsgBox "Code = " & ActiveDocument.Fields(1).Code & vbCrLf _
        & "Result = " & ActiveDocument.Fields(1).Result & vbCrLf
End If
```

Use the [Add](#) method to add a field to the **Fields** collection. The following example inserts a DATE field at the beginning of the selection and then displays the result.

```
Selection.Collapse Direction:=wdCollapseStart
Set myField = ActiveDocument.Fields.Add(Range:=Selection.Range, _
    Type:=wdFieldDate)
MsgBox myField.Result
```

The **wdFieldDate** constant is part of the **WdFieldType** group of constants, which includes all the various field types.



Fields Collection Object

Multiple objects [Fields \(Field\)](#)
└ Multiple objects

A collection of [Field](#) objects that represent all the fields in a selection, range, or document.

Using the Fields Collection

Use the **Fields** property to return the **Fields** collection. The following example updates all the fields in the selection.

```
Selection.Fields.Update
```

Use the [Add](#) method to add a field to the **Fields** collection. The following example inserts a DATE field at the beginning of the selection and then displays the result.

```
Selection.Collapse Direction:=wdCollapseStart
Set myField = ActiveDocument.Fields.Add(Range:=Selection.Range, _
    Type:=wdFieldDate)
MsgBox myField.Result
```

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 document. The following example displays the field code and the result of the first field in the active document.

```
If ActiveDocument.Fields.Count >= 1 Then
    MsgBox "Code = " & ActiveDocument.Fields(1).Code & vbCrLf _
        & "Result = " & ActiveDocument.Fields(1).Result & vbCrLf
End If
```

Remarks

Use the **Fields** property with a **MailMerge** object to return the **MailMergeFields** collection.

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.



FileConverter Object

[Application](#) | [FileConverters \(FileConverter\)](#)

Represents a file converter that's used to open or save files. The **FileConverter** object is a member of the [FileConverters](#) collection. The **FileConverters** collection contains all the installed file converters for opening and saving files.

Using the FileConverter Object

Use **FileConverters**(*index*), where *index* is a class name or index number, to return a single **FileConverter** object. The following example displays the extensions associated with the Microsoft Excel worksheet converter.

```
MsgBox FileConverters("MSBiff").Extensions
```

The index number represents the position of the file converter in the **FileConverters** collection. The following example displays the format name of the first file converter.

```
MsgBox FileConverters(1).FormatName
```

You cannot create a new file converter or add one to the **FileConverters** collection. **FileConverter** objects are added during installation of Microsoft Office or by installing supplemental file converters. Use either the [CanSave](#) or [CanOpen](#) property to determine whether a **FileConverter** object can be used to open or save document.

Remarks

File converters for saving documents are listed in the **Save As** dialog box. File converters for opening documents appear in a dialog box if the **Confirm conversion at Open** check box is selected on the **General** tab in the **Options** dialog box (**Tools** menu).

--

FileConverters Collection Object

[Application](#) | [FileConverters \(FileConverter\)](#)

A collection of **FileConverter** objects that represent all the file converters available for opening and saving files.

Using the FileConverters Collection

Use the **FileConverters** property to return the **FileConverters** collection. The following example determines whether a WordPerfect 6.0 converter is available.

```
For Each conv In FileConverters
    If conv.FormatName = "WordPerfect 6.x" Then
        MsgBox "WordPerfect 6.0 converter is installed"
    End if
Next conv
```

The **Add** method isn't available for the **FileConverters** collection. **FileConverter** objects are added during installation of Microsoft Office or by installing supplemental converters.

Use **FileConverters(index)**, where *index* is a class name or index number, to return a single **FileConverter** object. The following example displays the extensions associated with the Microsoft Excel worksheet converter.

```
MsgBox FileConverters("MSBiff").Extensions
```

The index number represents the position of the file converter in the **FileConverters** collection. The following example displays the format name of the first file converter.

```
MsgBox FileConverters(1).FormatName
```

Remarks

File converters for saving documents are listed in the **Save As** dialog box. File converters for opening documents appear in a dialog box if the **Confirm conversion at Open** check box is selected on the **General** tab in the **Options** dialog box (**Tools** menu).

--

FillFormat Object

[Shapes \(Shape\)](#) └ [FillFormat](#)
└ [ColorFormat](#)

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

Using the FillFormat Object

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

```
With ActiveDocument.Shapes _  
    .AddShape(msoShapeRectangle, 90, 90, 90, 80).Fill  
    .ForeColor.RGB = RGB(0, 128, 128)  
    .OneColorGradient msoGradientHorizontal, 1, 1  
End With
```

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.

--

Find Object

Multiple objects [Find](#)
└ Multiple objects

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

Using the Find Object

Use the **Find** property to return a **Find** object. The following example finds and selects the next occurrence of the word "hi."

```
With Selection.Find
    .ClearFormatting
    .Text = "hi"
    .Execute Forward:=True
End With
```

The following example finds all occurrences of the word "hi" in the active document and replaces the word with "hello."

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="hi", ReplaceWith:="hello", _
    Replace:=wdReplaceAll
```

Remarks

If you've gotten to the **Find** object from the **Selection** object, the selection is changed when text matching the find criteria is found. The following example selects the next occurrence of the word "blue."

```
Selection.Find.Execute FindText:="blue", Forward:=True
```

If you've gotten to the **Find** object from the **Range** object, the selection isn't changed when text matching the find criteria is found, but the **Range** object is redefined. The following example locates the first occurrence of the word "blue" in the active document. If "blue" is found in the document, myRange is redefined and bold formatting is applied to "blue."

```
Set myRange = ActiveDocument.Content  
myRange.Find.Execute FindText:="blue", Forward:=True  
If myRange.Find.Found = True Then myRange.Bold = True
```



FirstLetterException Object

[Application](#)  [AutoCorrect](#)
 [FirstLetterExceptions \(FirstLetterException\)](#)

Represents an abbreviation excluded from automatic correction. The **FirstLetterException** object is a member of the [FirstLetterExceptions](#) collection. The **FirstLetterExceptions** collection includes all the excluded abbreviations.

Note The first character following a period is automatically capitalized when the [CorrectSentenceCaps](#) property is set to **True**. The character you type following an item in the **FirstLetterExceptions** collection isn't capitalized.

Using the FirstLetterException Object

Use **FirstLetterExceptions**(*index*), where *index* is the abbreviation or the index number, to return a single **FirstLetterException** object. The following example deletes the abbreviation "appt." from the **FirstLetterExceptions** collection.

```
AutoCorrect.FirstLetterExceptions("appt.").Delete
```

The following example displays the name of the first item in the **FirstLetterExceptions** collection.

```
MsgBox AutoCorrect.FirstLetterExceptions(1).Name
```

Use the [Add](#) method to add an abbreviation to the list of first-letter exceptions. The following example adds the abbreviation "addr." to this list.

```
AutoCorrect.FirstLetterExceptions.Add Name:="addr."
```



FirstLetterExceptions Collection Object

[Application](#) └ [AutoCorrect](#)
└ [FirstLetterExceptions \(FirstLetterException\)](#)

A collection of [FirstLetterException](#) objects that represent the abbreviations excluded from automatic correction.

Note The first character following a period is automatically capitalized when the [CorrectSentenceCaps](#) property is set to **True**. The **FirstLetterExceptions** collection includes exceptions to this behavior (for example, abbreviations such as "addr." and "apt.").

Using the FirstLetterExceptions Collection

Use the **FirstLetterExceptions** property to return the **FirstLetterExceptions** collection. The following example deletes the abbreviation "addr." if it's included in the **FirstLetterExceptions** collection.

```
For Each aExcept In AutoCorrect.FirstLetterExceptions
    If aExcept.Name = "addr." Then aExcept.Delete
Next aExcept
```

The following example creates a new document and inserts all the AutoCorrect first-letter exceptions into it.

```
Documents.Add
For Each aExcept In AutoCorrect.FirstLetterExceptions
    With Selection
        .InsertAfter aExcept.Name
        .InsertParagraphAfter
        .Collapse Direction:=wdCollapseEnd
    End With
Next aExcept
```

Use the [Add](#) method to add an abbreviation to the list of first-letter exceptions. The following example adds the abbreviation "addr." to this list.

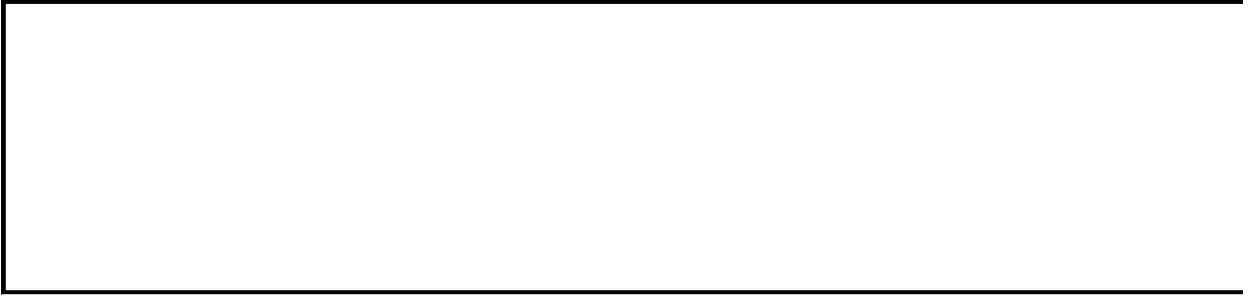
```
AutoCorrect.FirstLetterExceptions.Add Name:="addr."
```

Use **FirstLetterExceptions(index)**, where *index* is the abbreviation or the index number, to return a single **FirstLetterException** object. The following example deletes the abbreviation "appt." from the **FirstLetterExceptions** collection.

```
AutoCorrect.FirstLetterExceptions("appt.").Delete
```

The following example displays the name of the first item in the **FirstLetterExceptions** collection.

```
MsgBox AutoCorrect.FirstLetterExceptions(1).Name
```



Font Object

Multiple objects  [Font](#)
 Multiple objects

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.

```
Selection.Font.Bold = True
```

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

```
Set myRange = ActiveDocument.Paragraphs(1).Range
With myRange.Font
    .Bold = True
    .Name = "Arial"
    .Size = 24
End With
```

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

```
With ActiveDocument.Styles(wdStyleHeading2).Font
    .Name = "Arial"
    .Italic = True
End With
```

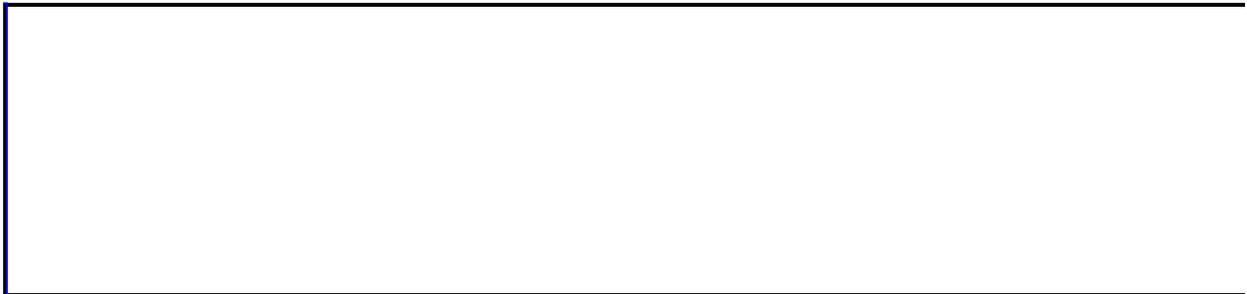
Remarks

You can use the **New** keyword to create a new, stand-alone **Font** object. The following example creates a **Font** object, sets some formatting properties, and then applies the **Font** object to the first paragraph in the active document.

```
Set myFont = New Font
myFont.Bold = True
myFont.Name = "Arial"
ActiveDocument.Paragraphs(1).Range.Font = myFont
```

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.

```
Set aFont = Selection.Font.Duplicate
aFont.Italic = True
ActiveDocument.Styles.Add(Name:="Italics", _
    Type:=wdStyleTypeCharacter).Font = aFont
```



FontNames Object

[Application](#) | [FontNames](#)

Represents a list of the names of all the available fonts.

Using the FontNames Object

Use the **FontNames**, **LandscapeFontNames**, or **PortraitFontNames** property to return the **FontNames** object. The following example displays the number of portrait fonts available.

```
MsgBox PortraitFontNames.Count & " fonts available"
```

This example lists all the font names in the **FontNames** object at the end of the active document.

```
For Each aFont In FontNames
    ActiveDocument.Range.InsertAfter aFont & vbCrLf
Next aFont
```

Use **FontNames(index)**, where *index* is the index number, to return the name of a font. The following example displays the first font name in the **FontNames** object.

```
MsgBox FontNames(1)
```

Remarks

You cannot add names to or remove names from the list of available font names.

--

Footnote Object

Multiple objects [Footnotes \(Footnote\)](#)
[Range](#)

Represents a footnote positioned at the bottom of the page or beneath text. The **Footnote** object is a member of the [Footnotes](#) collection. The **Footnotes** collection represents the footnotes in a selection, range, or document.

Using the Footnote Object

Use **Footnotes**(*index*), where *index* is the index number, to return a single **Footnote** object. The index number represents the position of the footnote in the selection, range, or document. The following example applies red formatting to the first footnote in the selection.

```
If Selection.Footnotes.Count >= 1 Then
    Selection.Footnotes(1).Reference.Font.ColorIndex = wdRed
End If
```

Use the [Add](#) method to add a footnote to the **Footnotes** collection. The following example inserts an automatically numbered footnote immediately after the selection.

```
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.Footnotes.Add Range:=Selection.Range , _
    Text:="The Willow Tree, (Lone Creek Press, 1996)."
```

Remarks

Footnotes positioned at the end of a document or section are considered endnotes and are included in the [Endnotes](#) collection.

FootnoteOptions Object

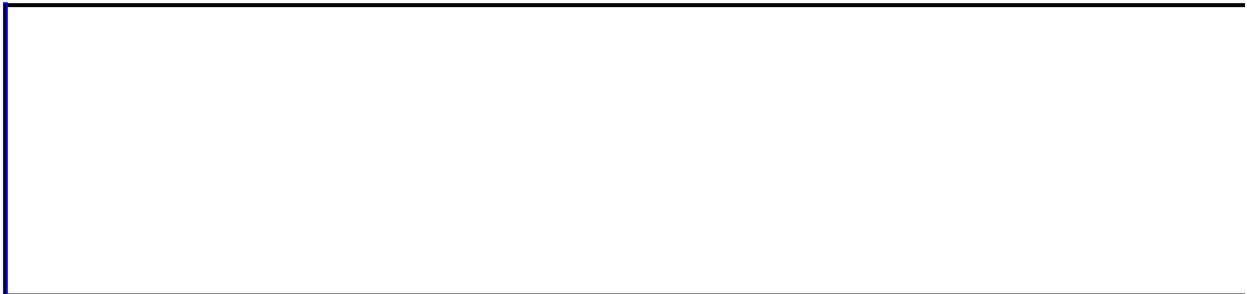
Multiple objects [FootnoteOptions](#)

Represents the properties assigned to a range or selection of footnotes in a document.

Using the FootnoteOptions object

Use the [Range](#) or [Selection](#) object to return a **FootnoteOptions** object. Using the **FootnoteOptions** object, you can assign different footnote properties to different areas of a document. For example, you may want footnotes in the introduction of a long document to be displayed as lowercase letters, while in the rest of your document they are displayed as asterisks. The following example uses the [NumberingRule](#), [NumberStyle](#), and [StartingNumber](#) properties to format the footnotes in the first section of the active document.

```
Sub BookIntro()  
    Dim rngIntro As Range  
  
    'Sets the range as section one of the active document  
    Set rngIntro = ActiveDocument.Sections(1).Range  
  
    'Formats the EndnoteOptions properties  
    With rngIntro.FootnoteOptions  
        .NumberingRule = wdRestartPage  
        .NumberStyle = wdNoteNumberStyleLowercaseLetter  
        .StartingNumber = 1  
    End With  
End Sub
```



Footnotes Collection Object

Multiple objects [Footnotes \(Footnote\)](#)
[Range](#)

A collection of [Footnote](#) objects that represent all the footnotes in a selection, range, or document.

Using the Footnotes Collection

Use the **Footnotes** property to return the **Footnotes** collection. The following example changes all of the footnotes in the active document to endnotes.

```
ActiveDocument.Footnotes.SwapWithEndnotes
```

Use the [Add](#) method to add a footnote to the **Footnotes** collection. The following example adds a footnote immediately after the selection.

```
Selection.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Footnotes.Add Range:=Selection.Range , _  
    Text:="The Willow Tree, (Lone Creek Press, 1996)."
```

Use **Footnotes(index)**, where *index* is the index number, to return a single **Footnote** object. The index number represents the position of the footnote in the selection, range, or document. The following example applies red formatting to the first footnote in the selection.

```
If Selection.Footnotes.Count >= 1 Then  
    Selection.Footnotes(1).Reference.Font.ColorIndex = wdRed  
End If
```

Remarks

Footnotes positioned at the end of a document or section are considered endnotes and are included in the [Endnotes](#) collection.

FormField Object

Multiple objects [└ FormFields \(FormField\)](#)
└ Multiple objects

Represents a single form field. The **FormField** object is a member of the [FormFields](#) collection.

Using the FormField Object

Use **FormFields**(*index*), where *index* is a bookmark name or index number, to return a single **FormField** object. The following example sets the result of the Text1 form field to "Don Funk."

```
ActiveDocument.FormFields("Text1").Result = "Don Funk"
```

The index number represents the position of the form field in the selection, range, or document. The following example displays the name of the first form field in the selection.

```
If Selection.FormFields.Count >= 1 Then  
    MsgBox Selection.FormFields(1).Name  
End If
```

Use the [Add](#) method with the **FormFields** object to add a form field. The following example adds a check box at the beginning of the active document and then selects the check box.

```
Set ffield = ActiveDocument.FormFields.Add( _  
    Range:=ActiveDocument.Range(Start:=0, End:=0), _  
    Type:=wdFieldFormCheckBox)  
ffield.CheckBox.Value = True
```

Remarks

Use the [CheckBox](#), [DropDown](#), and [TextInput](#) properties with the **FormField** object to return the **CheckDown**, **DropDown**, and **TextInput** objects. The following example selects the check box named "Check1."

```
ActiveDocument.FormFields("Check1").CheckBox.Value = True
```



FormFields Collection Object

Multiple objects [FormFields \(FormField\)](#)
└ Multiple objects

A collection of [FormField](#) objects that represent all the form fields in a selection, range, or document.

Using the FormFields Collection

Use the **FormFields** property to return the **FormFields** collection. The following example counts the number of text box form fields in the active document.

```
For Each aField In ActiveDocument.FormFields
    If aField.Type = wdFieldFormTextInput Then count = count + 1
Next aField
MsgBox "There are " & count & " text boxes in this document"
```

Use the [Add](#) method with the **FormFields** object to add a form field. The following example adds a check box at the beginning of the active document and then selects the check box.

```
Set ffield = ActiveDocument.FormFields.Add( _
    Range:=ActiveDocument.Range(Start:=0,End:=0), _
    Type:=wdFieldFormCheckBox)
ffield.CheckBox.Value = True
```

Use **FormFields(index)**, where *index* is a bookmark name or index number, to return a single **FormField** object. The following example sets the result of the Text1 form field to "Don Funk."

```
ActiveDocument.FormFields("Text1").Result = "Don Funk"
```

The index number represents the position of the form field in the selection, range, or document. The following example displays the name of the first form field in the selection.

```
If Selection.FormFields.Count >= 1 Then
    MsgBox Selection.FormFields(1).Name
End If
```



Frame Object

Multiple objects [└ Frames \(Frame\)](#)
└ Multiple objects

Represents a frame. The **Frame** object is a member of the [Frames](#) collection. The **Frames** collection includes all frames in a selection, range, or document.

Using the Frame Object

Use **Frames**(*index*), where *index* is the index number, to return a single **Frame** object. The index number represents the position of the frame in the selection, range, or document. The following example allows text to wrap around the first frame in the active document.

```
ActiveDocument.Frames(1).TextWrap = True
```

Use the **Add** method to add a frame around a range. The following example adds a frame around the first paragraph in the active document.

```
ActiveDocument.Frames.Add _  
    Range:=ActiveDocument.Paragraphs(1).Range
```

Remarks

You can wrap text around **Shape** or **ShapeRange** objects by using the [WrapFormat](#) property. You can position a **Shape** or **ShapeRange** object by using the [Top](#) and [Left](#) properties.

Frames Collection Object

Multiple objects [└ Frames \(Frame\)](#)
└ Multiple objects

A collection of [Frame](#) objects in a selection, range, or document.

Using the Frames Collection

Use the **Frames** property to return the **Frames** collection. The following example removes borders from all frames in the active document.

```
For Each aFrame In ActiveDocument.Frames
    aFrame.Borders.Enable = False
Next aFrame
```

Use the [Add](#) method to add a frame around a range. The following example adds a frame around the first paragraph in the active document.

```
ActiveDocument.Frames.Add _
    Range:=ActiveDocument.Paragraphs(1).Range
```

Use **Frames(index)**, where *index* is the index number, to return a single **Frame** object. The index number represents the position of the frame in the selection, range, or document. The following example causes text to wrap around the first frame in the first section of the active document.

```
ActiveDocument.Sections(1).Range.Frames(1).TextWrap = True
```

Remarks

You can wrap text around **Shape** or **ShapeRange** objects by using the [WrapFormat](#) property. You can position a **Shape** or **ShapeRange** object by using the [Top](#) and [Left](#) properties.

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.



Frameset Object

Multiple objects [└ Frameset](#)

Represents an entire frames page or a single frame on a frames page. There is no Framesets collection; each **Document** object or **Pane** object contains only one **Frameset** object.

Using the Frameset Object

Use the [Frameset](#) property to return the **Frameset** object. For properties or methods that affect all frames on a frames page, use the **Frameset** object from the **Document** object (`ActiveWindow.Document.Frameset`). For properties or methods that affect individual frames on a frames page, use the **Frameset** object from the **Pane** object (`ActiveWindow.ActivePane.Frameset`).

This example opens a file named "Proposal.doc," creates a frames page based on the file, and adds a frame (on the left side of the page) containing a table of contents for the file.

```
Documents.Open "C:\My Documents\proposal.doc"  
ActiveDocument.ActiveWindow.ActivePane.NewFrameset  
ActiveDocument.ActiveWindow.ActivePane.TOCInFrameset
```

This example adds a new frame to the right of the specified frame.

```
ActiveDocument.ActiveWindow.ActivePane.Frameset _  
    .AddNewFrame wdFramesetNewRight
```

This example sets the name of the third child **Frameset** object of the frames page to "BottomFrame."

```
ActiveWindow.Document.Frameset _  
    .ChildFramesetItem(3).FrameName = "BottomFrame"
```

This example links the specified frame to a local file called "Order.htm." It sets the frame to be resizable, to appear with scrollbars in a Web browser, and to be 25% as high as the active window.

```
With ActiveDocument.ActiveWindow.ActivePane.Frameset  
    .FrameDefaultURL = "C:\My Documents\order.htm"  
    .FrameLinkToFile = True  
    .FrameResizable = True  
    .FrameScrollbarType = wdScrollbarTypeYes  
    .HeightType = wdFramesetSizeTypePercent  
    .Height = 25  
End With
```

This example sets Microsoft Word to display frame borders in the specified

frames page.

```
ActiveDocument.ActiveWindow.ActivePane.Frameset _  
    .FrameDisplayBorders = True
```

This example sets the frame borders on the frames page to be 6 points wide and tan.

```
With ActiveWindow.Document.Frameset  
    .FramesetBorderColor = wdColorTan  
    .FramesetBorderWidth = 6  
End With
```

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

FreeformBuilder Object

Multiple objects [└FreeformBuilder](#)
[└Shape](#)

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

Using the FreeformBuilder Object

Use the [BuildFreeform](#) method 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.

```
With ActiveDocument.Shapes _  
    .BuildFreeform(msoEditingCorner, 360, 200)  
    .AddNodes msoSegmentCurve, msoEditingCorner, _  
        380, 230, 400, 250, 450, 300  
    .AddNodes msoSegmentCurve, msoEditingAuto, 480, 200  
    .AddNodes msoSegmentLine, msoEditingAuto, 480, 400  
    .AddNodes msoSegmentLine, msoEditingAuto, 360, 200  
    .ConvertToShape  
End With
```



Global Object

[Global](#) ↳ Multiple objects

Contains top-level properties and methods that don't need to be preceded by the **Application** property. For example, the following two statements have the same result.

```
Documents(1).Content.Bold = True  
Application.Documents(1).Content.Bold = True
```



GroupShapes Collection Object

[Shapes \(Shape\)](#) | [GroupShapes \(Shape\)](#)

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 the **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 second triangle only.

```
With ActiveDocument.Shapes
    .AddShape(msoShapeIsoscelesTriangle, _
        10, 10, 100, 100).Name = "shpOne"
    .AddShape(msoShapeIsoscelesTriangle, _
        150, 10, 100, 100).Name = "shpTwo"
    .AddShape(msoShapeIsoscelesTriangle, _
        300, 10, 100, 100).Name = "shpThree"
With .Range(Array("shpOne", "shpTwo", "shpThree")).Group
    .Fill.PresetTextured msoTextureBlueTissuePaper
    .GroupItems(2).Fill.PresetTextured msoTextureGreenMarble
End With
End With
```



HangulAndAlphabetException Object

[HangulAndAlphabetExceptions](#) | [HangulAndAlphabetException](#)

Represents a single Hangul or alphabet AutoCorrect exception. The **HangulAndAlphabetException** object is a member of the [HangulAndAlphabetExceptions](#) collection. The **HangulAndAlphabetExceptions** collection includes all Hangul and alphabet AutoCorrect exceptions and corresponds to the items listed on the **Korean** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu).

Using the HangulAndAlphabetException Object

Use **HangulAndAlphabetExceptions**(*index*), where *index* is the Hangul or alphabet AutoCorrect exception name or the index number, to return a single **HangulAndAlphabetException** object. The following example deletes the alphabet AutoCorrect exception named "hello."

```
AutoCorrect.HangulAndAlphabetExceptions("hello").Delete
```

The index number represents the position of the Hangul or alphabet AutoCorrect exception in the **HangulAndAlphabetExceptions** collection. The following example displays the name of the first item in the **HangulAndAlphabetExceptions** collection.

```
MsgBox AutoCorrect.HangulAndAlphabetExceptions(1).Name
```

If the value of the [HangulAndAlphabetAutoAdd](#) property is **True**, words are automatically added to the list of Hangul and alphabet AutoCorrect exceptions. Use the [Add](#) method to add an item to the **HangulAndAlphabetExceptions** collection. The following example adds "goodbye" to the list of alphabet AutoCorrect exceptions.

```
AutoCorrect.HangulAndAlphabetExceptions.Add Name:="goodbye"
```

Remarks

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

--

HangulAndAlphabetExceptions Collection Object

[AutoCorrect](#) └ [HangulAndAlphabetExceptions](#)
└ [HangulAndAlphabetException](#)

A collection of [HangulAndAlphabetException](#) objects that represents all Hangul and alphabet AutoCorrect exceptions. This list corresponds to the list of AutoCorrect exceptions on the **Korean** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu).

Using the HangulAndAlphabetExceptions Collection

Use the [HangulAndAlphabetExceptions](#) property to return the **HangulAndAlphabetExceptions** collection. The following example displays the items in this collection.

```
For Each aHan In AutoCorrect.HangulAndAlphabetExceptions
    MsgBox aHan.Name
Next aHan
```

If the value of the [HangulAndAlphabetAutoAdd](#) property is **True**, words are automatically added to the list of Hangul and alphabet AutoCorrect exceptions. Use the [Add](#) method to add an item to the **HangulAndAlphabetExceptions** collection. The following example adds "hello" to the list of alphabet AutoCorrect exceptions.

```
AutoCorrect.HangulAndAlphabetExceptions.Add Name:="hello"
```

Use **HangulAndAlphabetExceptions(index)**, where *index* is the Hangul or alphabet AutoCorrect exception name or the index number, to return a single **HangulAndAlphabetException** object. The following example deletes the alphabet AutoCorrect exception named "goodbye."

```
AutoCorrect.HangulAndAlphabetExceptions("goodbye").Delete
```

The index number represents the position of the hangul or alphabet AutoCorrect exception in the **HangulAndAlphabetExceptions** collection. The following example displays the name of the first item in the **HangulAndAlphabetExceptions** collection.

```
MsgBox AutoCorrect.HangulAndAlphabetExceptions(1).Name
```

Remarks

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

--

HangulHanjaConversionDictionaries Collection Object

Multiple objects [└ HangulHanjaConversionDictionaries](#)
[└ Dictionary](#)

A collection of [Dictionary](#) objects that includes the active custom Hangul-Hanja conversion dictionaries.

Using the HangulHanjaConversionDictionaries Collection

Use the [HangulHanjaDictionaries](#) property to return the collection of currently active custom conversion dictionaries. The following example displays the names of all the active custom conversion dictionaries.

```
For Each d In HangulHanjaDictionaries
    MsgBox d.Name
Next d
```

Use the [Add](#) method to add a new custom conversion dictionary to the collection of active custom conversion dictionaries. If there isn't a file with the name specified by *FileName*, Microsoft Word creates it. The following example adds "Hanja1.hhd" to the collection of custom conversion dictionaries.

```
CustomDictionaries.Add FileName:="Hanja1.hhd"
```

Use the [ClearAll](#) method to unload all custom conversion dictionaries. Note, however, that this method doesn't delete the dictionary files. After you use this method, the number of custom conversion dictionaries in the collection is 0 (zero). The following example clears the custom conversion dictionaries and creates a new custom conversion dictionary file. The new dictionary is set as the active custom dictionary to which Word will automatically add any new words it encounters.

```
With HangulHanjaDictionaries
    .ClearAll
    .Add FileName:= "Hanja1.hhd"
    .ActiveCustomDictionary = HangulHanjaDictionaries(1)
End With
```

Remarks

You set the custom dictionary to which new words are added by using the [ActiveCustomDictionary](#) property. If you try to set this property to a dictionary that isn't a custom conversion dictionary, an error occurs.

The [Maximum](#) property returns the maximum number of simultaneous custom conversion dictionaries that the application can support. For Word, this maximum is 10.

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).



HeaderFooter Object

Multiple objects [HeaderFooter](#)
└ Multiple objects

Represents a single header or footer. The **HeaderFooter** object is a member of the [HeadersFooters](#) collection. The **HeadersFooters** collection includes all headers and footers in the specified document section.

Using the HeaderFooter Object

Use **Headers**(*index*) or **Footers**(*index*), where *index* is one of the **WdHeaderFooterIndex** constants (**wdHeaderFooterEvenPages**, **wdHeaderFooterFirstPage**, or **wdHeaderFooterPrimary**), to return a single **HeaderFooter** object. The following example changes the text of both the primary header and the primary footer in the first section of the active document.

```
With ActiveDocument.Sections(1)
    .Headers(wdHeaderFooterPrimary).Range.Text = "Header text"
    .Footers(wdHeaderFooterPrimary).Range.Text = "Footer text"
End With
```

You can also return a single **HeaderFooter** object by using the [HeaderFooter](#) property with a [Selection](#) object.

Note You cannot add **HeaderFooter** objects to the **HeadersFooters** collection.

Remarks

Use the [DifferentFirstPageHeaderFooter](#) property with the [PageSetup](#) object to specify a different first page. The following example inserts text into the first page footer in the active document.

```
With ActiveDocument
    .PageSetup.DifferentFirstPageHeaderFooter = True
    .Sections(1).Footers(wdHeaderFooterFirstPage) _
        .Range.InsertBefore _
            "Written by Joe Smith"
End With
```

Use the [OddAndEvenPagesHeaderFooter](#) property with the [PageSetup](#) object to specify different odd and even page headers and footers. If the [OddAndEvenPagesHeaderFooter](#) property is **True**, you can return an odd header or footer by using **wdHeaderFooterPrimary**, and you can return an even header or footer by using **wdHeaderFooterEvenPages**.

Use the [Add](#) method with the [PageNumbers](#) object to add a page number to a header or footer. The following example adds page numbers to the primary footer in the first section of the active document.

```
With ActiveDocument.Sections(1)
    .Footers(wdHeaderFooterPrimary).PageNumbers.Add
End With
```



HeadersFooters Collection Object

[Sections \(Section\)](#)  [HeadersFooters \(HeaderFooter\)](#)

 Multiple objects

A collection of [HeaderFooter](#) objects that represent the headers or footers in the specified section of a document.

Using the HeadersFooters Collection

Use the **Headers** or **Footers** property to return the **HeadersFooters** collection. The following example displays the text from the primary footer in the first section of the active document.

```
With ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)
    If .Range.Text <> vbCr Then
        MsgBox .Range.Text
    Else
        MsgBox "Footer is empty"
    End If
End With
```

Note You cannot add **HeaderFooter** objects to the **HeadersFooters** collection.

Use **Headers(index)** or **Footers(index)**, where *index* is one of the **WdHeaderFooterIndex** constants (**wdHeaderFooterEvenPages**, **wdHeaderFooterFirstPage**, or **wdHeaderFooterPrimary**), to return a single **HeaderFooter** object. The following example changes the text of both the primary header and the primary footer the first section of the active document.

```
With ActiveDocument.Sections(1)
    .Headers(wdHeaderFooterPrimary).Range.Text = "Header text"
    .Footers(wdHeaderFooterPrimary).Range.Text = "Footer text"
End With
```

You can also return a single **HeaderFooter** object by using the [HeaderFooter](#) property with a **Selection** object.

Remarks

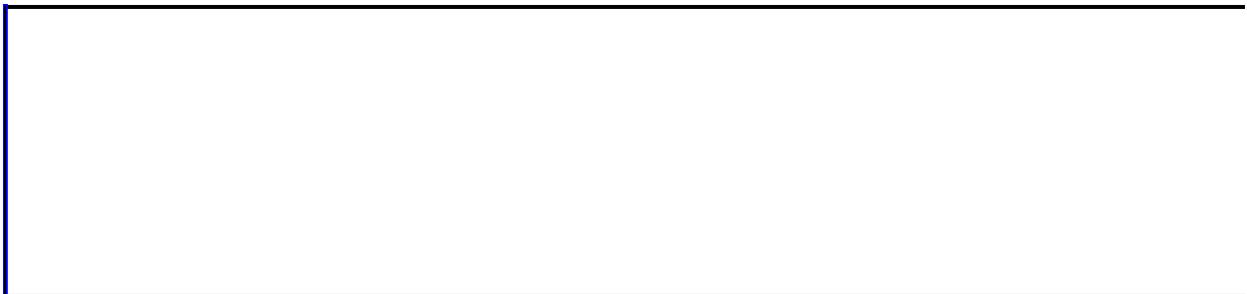
Use the [DifferentFirstPageHeaderFooter](#) property with the **PageSetup** object to specify a different first page. The following example inserts text into the first page footer in the active document.

```
With ActiveDocument
    .PageSetup.DifferentFirstPageHeaderFooter = True
    .Sections(1).Footers(wdHeaderFooterFirstPage) _
        .Range.InsertBefore _
            "Written by Kate Edson"
End With
```

Use the [OddAndEvenPagesHeaderFooter](#) property with the **PageSetup** object to specify different odd and even page headers and footers. If the **OddAndEvenPagesHeaderFooter** property is **True**, you can return an odd header or footer by using **wdHeaderFooterPrimary**, and you can return an even header or footer by using **wdHeaderFooterEvenPages**.

Use the [Add](#) method with the **PageNumbers** object to add a page number to a header or footer. The following example adds page numbers to the first page footer in the first section in the active document.

```
With ActiveDocument.Sections(1)
    .PageSetup.DifferentFirstPageHeaderFooter = True
    .Footers(wdHeaderFooterPrimary).PageNumbers.Add _
        FirstPage:=True
End With
```



HeadingStyle Object

[Documents \(Document\)](#) └ Multiple objects
└ [HeadingStyles \(HeadingStyle\)](#)

Represents a style used to build a table of contents or figures. The **HeadingStyle** object is a member of the [HeadingStyles](#) collection.

Using the HeadingStyle Object

Use **HeadingStyles**(*index*), where *index* is the index number, to return a single **HeadingStyle** object. The index number represents the position of the style in the **HeadingStyles** collection. The following example adds (at the beginning of the active document) a table of figures built from the Title style, and then displays the name of the first style in the **HeadingStyles** collection.

```
Set myTOF = ActiveDocument.TablesOfFigures.Add _  
    (Range:=ActiveDocument.Range(0, 0), AddedStyles:="Title")  
MsgBox myTOF.HeadingStyles(1).Style
```

Use the [Add](#) method to add a style to the **HeadingStyles** collection. The following example adds a table of contents at the beginning of the active document and then adds the Title style to the list of styles used to build a table of contents.

```
Set myToc = ActiveDocument.TablesOfContents.Add _  
    (Range:=ActiveDocument.Range(0, 0), UseHeadingStyles:=True, _  
    LowerHeadingLevel:=3, UpperHeadingLevel:=1)  
myToc.HeadingStyles.Add Style:="Title", Level:=2
```



HeadingStyles Collection Object

[Documents \(Document\)](#) └ Multiple objects
└ [HeadingStyles \(HeadingStyle\)](#)

A collection of [HeadingStyle](#) objects that represent the styles used to compile a table of figures or table of contents.

Using the HeadingStyles Collection

Use the **HeadingStyles** property to return the **HeadingStyles** collection. The following example displays the number of items in the **HeadingStyles** collection for the first table of contents in the active document.

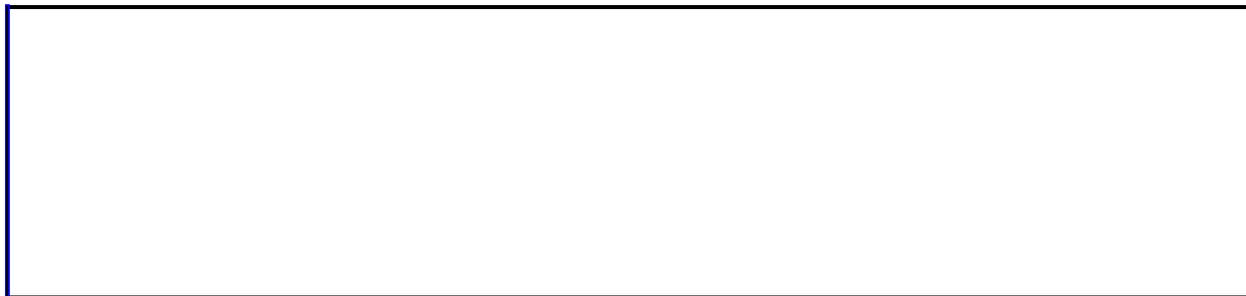
```
MsgBox ActiveDocument.TablesOfContents(1).HeadingStyles.Count
```

Use the [Add](#) method to add a style to the **HeadingStyles** collection. The following example adds a table of contents at the beginning of the active document and then adds the Title style to the list of styles used to build a table of contents.

```
Set myToc = ActiveDocument.TablesOfContents.Add _  
    (Range:=ActiveDocument.Range(0, 0), UseHeadingStyles:=True, _  
    LowerHeadingLevel:=3, UpperHeadingLevel:=1)  
myToc.HeadingStyles.Add Style:="Title", Level:=2
```

Use **HeadingStyles(index)**, where *index* is the index number, to return a single **HeadingStyle** object. The index number represents the position of the style in the **HeadingStyles** collection. The following example adds (at the beginning of the active document) a table of figures built from the Title style, and then displays the name of the first style in the **HeadingStyles** collection.

```
Set myTOF = ActiveDocument.TablesOfFigures.Add _  
    (Range:=ActiveDocument.Range(0, 0), AddedStyles:="Title")  
MsgBox myTOF.HeadingStyles(1).Style
```



HorizontalLineFormat Object

[InlineShapes \(InlineShape\)](#)  [HorizontalLineFormat](#)

Represents horizontal line formatting.

Using the HorizontalLineFormat Object

Use the [HorizontalLineFormat](#) property to return a **HorizontalLineFormat** object. This example sets the alignment for a new horizontal line.

```
Selection.InlineShapes.AddHorizontalLineStandard  
ActiveDocument.InlineShapes(1) _  
    .HorizontalLineFormat.Alignment = _  
    wdHorizontalLineAlignLeft
```

This example adds a horizontal line without any 3-D shading.

```
Selection.InlineShapes.AddHorizontalLineStandard  
ActiveDocument.InlineShapes(1) _  
    .HorizontalLineFormat.NoShade = True
```

This example adds a horizontal line and sets its length to 50% of the window width.

```
Selection.InlineShapes.AddHorizontalLineStandard  
ActiveDocument.InlineShapes(1) _  
    .HorizontalLineFormat.PercentWidth = 50
```



HTMLDivision Object

[HTMLDivisions](#)  [HTMLDivision](#)
 Multiple objects

Represents a single HTML division that can be added to a Web document. The **HTMLDivision** object is a member of the [HTMLDivisions](#) collection.

Using the HTMLDivision object

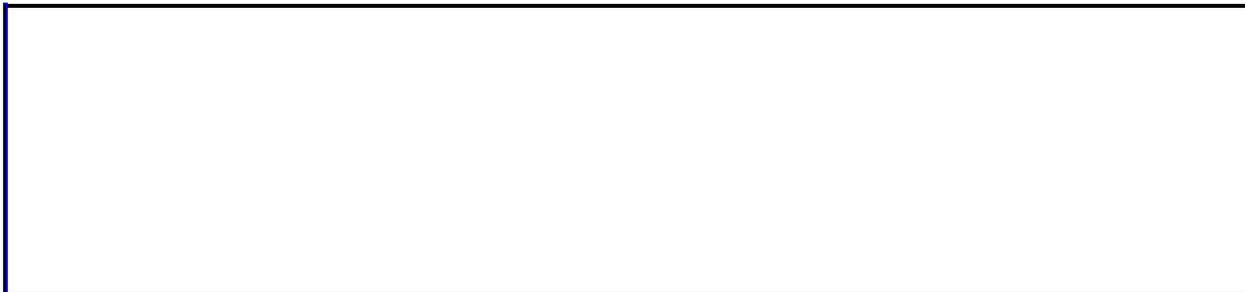
Use [HTMLDivisions\(index\)](#), where *index* refers to the HTML division in the document, to return a single **HTMLDivision** object. Use the [Borders](#) property to format border properties for an HTML division. This example formats three nested divisions in the active document. This example assumes that the active document is an HTML document with at least three divisions.

```
Sub FormatHTMLDivisions()  
  With ActiveDocument.HTMLDivisions(1)  
    With .Borders(wdBorderLeft)  
      .Color = wdColorRed  
      .LineStyle = wdLineStyleSingle  
    End With  
    With .Borders(wdBorderTop)  
      .Color = wdColorRed  
      .LineStyle = wdLineStyleSingle  
    End With  
    With .HTMLDivisions(1)  
      .LeftIndent = InchesToPoints(1)  
      .RightIndent = InchesToPoints(1)  
      With .Borders(wdBorderRight)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
    End With  
    With .Borders(wdBorderBottom)  
      .Color = wdColorBlue  
      .LineStyle = wdLineStyleDouble  
    End With  
    With .HTMLDivisions(1)  
      .LeftIndent = InchesToPoints(1)  
      .RightIndent = InchesToPoints(1)  
      With .Borders(wdBorderLeft)  
        .Color = wdColorBlack  
        .LineStyle = wdLineStyleDot  
      End With  
      With .Borders(wdBorderTop)  
        .Color = wdColorBlack  
        .LineStyle = wdLineStyleDot  
      End With  
    End With  
  End With  
End With
```

End Sub

HTML divisions can be nested within multiple HTML divisions. Use the [HTMLDivisionParent](#) method to access a parent HTML division of the current HTML division. This example formats the borders for two HTML divisions in the active document. This example assumes that the active document is an HTML document with at least two divisions.

```
Sub FormatHTMLDivisions()  
  With ActiveDocument.HTMLDivisions(1)  
    With .HTMLDivisions(1)  
      .LeftIndent = InchesToPoints(1)  
      .RightIndent = InchesToPoints(1)  
      With .Borders(wdBorderLeft)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .Borders(wdBorderRight)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .HTMLDivisionParent  
        .LeftIndent = InchesToPoints(1)  
        .RightIndent = InchesToPoints(1)  
        With .Borders(wdBorderTop)  
          .Color = wdColorBlack  
          .LineStyle = wdLineStyleDot  
        End With  
        With .Borders(wdBorderBottom)  
          .Color = wdColorBlack  
          .LineStyle = wdLineStyleDot  
        End With  
      End With  
    End With  
  End With  
End Sub
```



HTMLDivisions Collection

Multiple objects [HTMLDivisions](#)
[HTMLDivision](#)

A collection of [HTMLDivision](#) objects that represents the HTML divisions that exist in a Web document.

Using the HTMLDivisions collection

Use the [HTMLDivisions](#) property to return the **HTMLDivisions** collection. Use the [Add](#) method to add an HTML division to a Web document. This example adds a new HTML division to the active document, adds text to the division, and formats the borders around the division.

```
Sub NewDivision()  
  
    With ActiveDocument.HTMLDivisions  
        .Add  
        .Item(Index:=1).Range.Text = "This is a new HTML division."  
        With .Item(1)  
            With .Borders(wdBorderBottom)  
                .LineStyle = wdLineStyleTriple  
                .LineWidth = wdLineWidth025pt  
                .Color = wdColorRed  
            End With  
            With .Borders(wdBorderTop)  
                .LineStyle = wdLineStyleDot  
                .LineWidth = wdLineWidth050pt  
                .Color = wdColorBlue  
            End With  
            With .Borders(wdBorderLeft)  
                .LineStyle = wdLineStyleDouble  
                .LineWidth = wdLineWidth075pt  
                .Color = wdColorBrightGreen  
            End With  
            With .Borders(wdBorderRight)  
                .LineStyle = wdLineStyleDashDotDot  
                .LineWidth = wdLineWidth075pt  
                .Color = wdColorTurquoise  
            End With  
        End With  
    End With  
End Sub
```



Hyperlink Object

Multiple objects [└ Hyperlinks \(Hyperlink\)](#)
└ Multiple objects

Represents a hyperlink. The **Hyperlink** object is a member of the [Hyperlinks](#) collection.

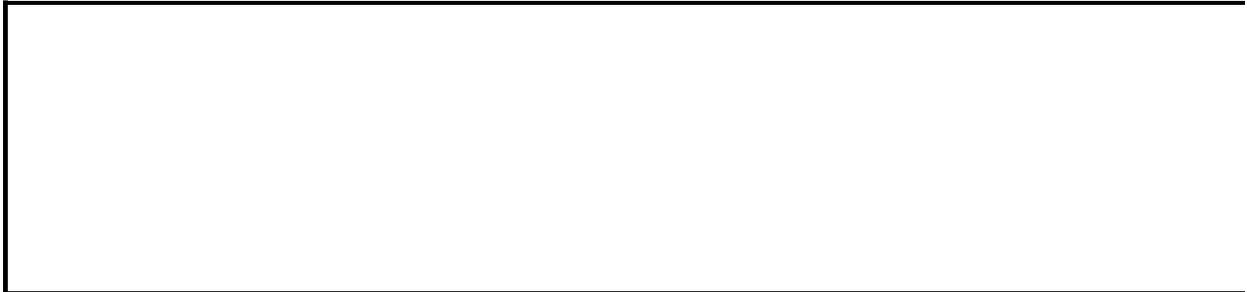
Using the Hyperlink Object

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

```
ActiveDocument.Shapes(1).Hyperlink.Follow
```

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

```
If Selection.HyperLinks.Count >= 1 Then  
    Selection.HyperLinks(1).Follow  
End If
```



Hyperlinks Collection Object

Multiple objects [Hyperlinks \(Hyperlink\)](#)
└ Multiple objects

Represents the collection of [Hyperlink](#) objects in a document, range, or selection.

Using the Hyperlinks Collection

Use the **Hyperlinks** property to return the **Hyperlinks** collection. The following example checks all the hyperlinks in document one for a link that contains the word "Microsoft" in the address. If a hyperlink is found, it's activated with the **Follow** method.

```
For Each hLink In Documents(1).Hyperlinks
    If InStr(hLink.Address, "Microsoft") <> 0 Then
        hLink.Follow
        Exit For
    End If
Next hLink
```

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

```
ActiveDocument.Hyperlinks.Add Address:="http://www.msn.com/", _
    Anchor:=Selection.Range
```

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

```
If Selection.HyperLinks.Count >= 1 Then
    Selection.HyperLinks(1).Follow
End If
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.

Index Object

[Indexes](#) | [Index](#)
| [Range](#)

Represents a single index. The **Index** object is a member of the [Indexes](#) collection. The **Indexes** collection includes all the indexes in the specified document.

Using the Index Object

Use **Indexes**(*index*), where *index* is the index number, to return a single **Index** object. The index number represents the position of the **Index** object in the document. The following example updates the first index in the active document.

```
If ActiveDocument.Indexes.Count >= 1 Then
    ActiveDocument.Indexes(1).Update
End If
```

Use the [Add](#) method to create an index and add it to the **Indexes** collection. The following example creates an index at the end of the active document.

```
Set myRange = ActiveDocument.Content
myRange.Collapse Direction:=wdCollapseEnd
ActiveDocument.Indexes.Add Range:=myRange, Type:=wdIndexRunin
```



Indexes Collection Object

[Document](#) └ [Indexes](#)
└ Multiple objects

A collection of [Index](#) objects that represents all the indexes in the specified document.

Using the Indexes Collection

Use the [Indexes](#) property to return the **Indexes** collection. The following example formats indexes in the active document with the classic format.

```
ActiveDocument.Indexes.Format = wdIndexClassic
```

Use the [Add](#) method to create an index and add it to the **Indexes** collection. The following example creates an index at the end of the active document.

```
Set myRange = ActiveDocument.Content  
myRange.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Indexes.Add Range:=myRange, Type:=wdIndexRunin
```

Use **Indexes(index)**, where *index* is the index number, to return a single **Index** object. The index number represents the position of the **Index** object in the document. The following example updates the first index in the active document.

```
If ActiveDocument.Indexes.Count >= 1 Then  
    ActiveDocument.Indexes(1).Update  
End If
```



InlineShape Object

Multiple objects [↳ InlineShapes \(InlineShape\)](#)
↳ Multiple objects

Represents an object in the text layer of a document. An inline shape can only be a picture, an OLE object, or an ActiveX control. **InlineShape** objects are treated like characters and are positioned as characters within a line of text. The **InlineShape** object is a member of the [InlineShapes](#) collection. The **InlineShapes** collection contains all the shapes in a document, range, or selection.

Using the InlineShape Object

Use **InlineShapes**(*index*), where *index* is the index number, to return a single **InlineShape** object. Inline shapes don't have names. The following example activates the first inline shape in the active document.

```
ActiveDocument.InlineShapes(1).Activate
```

Remarks

Shape objects are anchored to a range of text but are free-floating and can be positioned anywhere on the page. You can use the [ConvertToInlineShape](#) method and the [ConvertToShape](#) method to convert shapes from one type to the other. You can convert only pictures, OLE objects, and ActiveX controls to inline shapes. Use the [Type](#) property to return the type of inline shape: picture, linked picture, embedded OLE object, linked OLE object, or ActiveX control.

When you open a document created in an earlier version of Word, pictures are converted to inline shapes.



InlineShapes Collection Object

Multiple objects [↳ InlineShapes \(InlineShape\)](#)
↳ Multiple objects

A collection of [InlineShape](#) objects that represent all the inline shapes in a document, range, or selection.

Using the InlineShapes Collection

Use the **InlineShapes** property to return the **InlineShapes** collection. The following example converts each inline shape in the active document to a **Shape** object.

```
For Each iShape In ActiveDocument.InlineShapes
    iShape.ConvertToShape
Next iShape
```

Use the [New](#) method to create a new picture as an inline shape. You can use the [AddPicture](#) and [AddOLEObject](#) methods to add pictures or OLE objects and link them to a source file. Use the [AddOLEControl](#) method to add an ActiveX control.

Remarks

Shape objects are anchored to a range of text but are free-floating and can be positioned anywhere on the page. You can use the [ConvertToInlineShape](#) method and the [ConvertToShape](#) method to convert shapes from one type to the other. You can convert only pictures, OLE objects, and ActiveX controls to inline shapes.

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.

When you open a document created in an earlier version of Word, pictures are converted to inline shapes.



KeyBinding Object

Multiple objects [↳ KeyBindings \(KeyBinding\)](#)

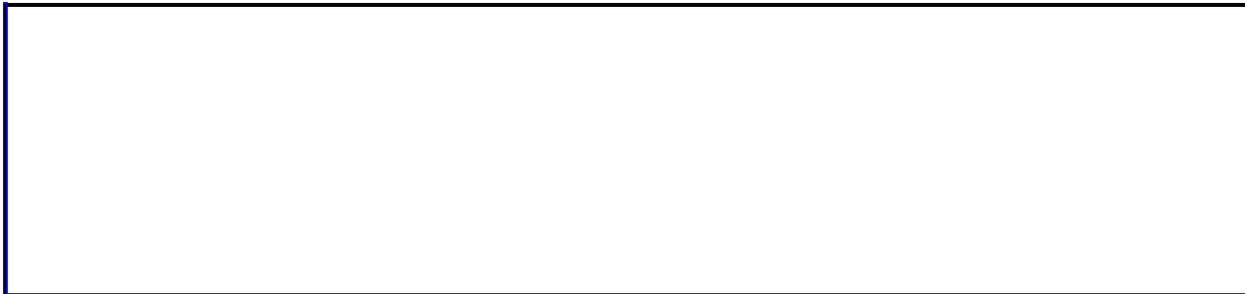
Represents a custom key assignment in the current context. The **KeyBinding** object is a member of the [KeyBindings](#) collection. Custom key assignments are made in the **Customize Keyboard** dialog box.

Using the KeyBinding Object

Use **KeyBindings**(*index*), where *index* is the index number, to return a single **KeyBinding** object. The following example displays the command associated with the first **KeyBinding** object in the **KeyBindings** collection.

```
MsgBox KeyBindings(1).Command
```

You can also use the [FindKey](#) property and the [Key](#) method to return a **KeyBinding** object.



KeyBindings Collection Object

[Application](#) | [KeyBindings \(KeyBinding\)](#)

A collection of [KeyBinding](#) objects that represent the custom key assignments in the current context. Custom key assignments are made in the **Customize Keyboard** dialog box.

Using the KeyBindings Collection

Use the **KeyBindings** property to return the **KeyBindings** collection. The following example inserts after the selection the command name and key combination for each item in the **KeyBindings** collection.

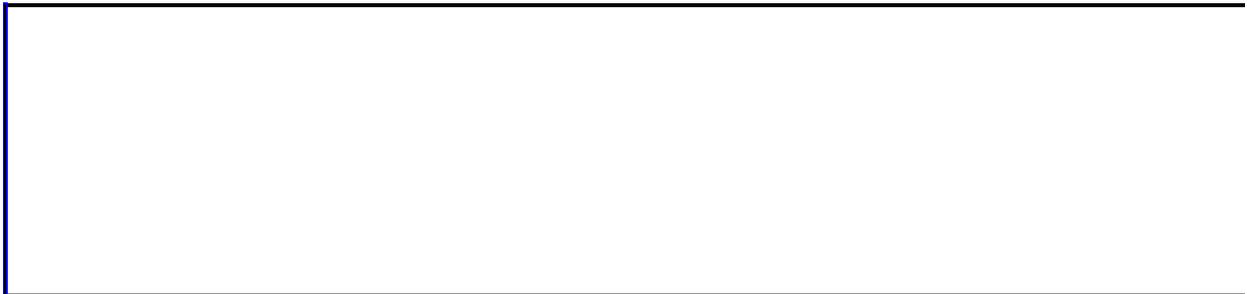
```
CustomizationContext = NormalTemplate
For Each aKey In KeyBindings
    Selection.InsertAfter aKey.Command & vbTab _
        & aKey.KeyString & vbCr
    Selection.Collapse Direction:=wdCollapseEnd
Next aKey
```

Use the [Add](#) method to add a **KeyBinding** object to the **KeyBindings** collection. The following example adds the CTRL+ALT+H key combination to the Heading 1 style in the active document.

```
CustomizationContext = ActiveDocument
KeyBindings.Add KeyCategory:=wdKeyCategoryStyle, _
    Command:="Heading 1", _
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyH)
```

Use **KeyBindings(index)**, where *index* is the index number, to return a single **KeyBinding** object. The following example displays the command associated with the first **KeyBinding** object in the **KeyBindings** collection.

```
MsgBox KeyBindings(1).Command
```



KeysBoundTo Collection Object

[Application](#)  [KeysBoundTo \(KeyBinding\)](#)

A collection of [KeyBinding](#) objects assigned to a command, style, macro, or other item in the current context.

Using the **KeysBoundTo** Collection

Use the **KeysBoundTo** property to return the **KeysBoundTo** collection. The following example displays the key combinations assigned to the **FileNew** command in the Normal template.

```
CustomizationContext = NormalTemplate
For Each myKey In KeysBoundTo(KeyCategory:=wdKeyCategoryCommand, _
    Command:="FileNew")
    myStr = myStr & myKey.KeyString & vbCr
Next myKey
MsgBox myStr
```

The following example displays the name of the document or template where the keys for the macro named "Macro1" are stored.

```
Set kb = KeysBoundTo(KeyCategory:=wdKeyCategoryMacro, _
    Command:="Macro1")
MsgBox kb.Context.Name
```



Language Object

[Languages](#) | [Language](#)
| [Dictionary](#)

Represents a language used for proofing or formatting in Microsoft Word. The **Language** object is a member of the [Languages](#) collection.

Using the Language object

Use **Languages**(*index*) to return a single **Language** object, where *index* can be the value of the **Name** property, the value of the **NameLocal** property, one of the **WdLanguageID** constants, or one of the **MsoLanguageID** constants. (For the list of valid **WdLanguageID** or **MsoLanguageID** constants, see the Object Browser in the Visual Basic Editor.)

The **Name** property returns the name of a language, whereas the **NameLocal** property returns the name of a language in the language of the user. The following example returns the string "Italiano" for **Name** and "Italian (Standard)" for **NameLocal** when it's run in the U.S. English version of Word.

```
Sub ShowItalianNames()  
    MsgBox Languages(wdItalian).Name  
    MsgBox Languages(wdItalian).NameLocal  
End Sub
```

Returning the Active Proofing Dictionaries

For each language for which proofing tools are installed, you can use the [ActiveGrammarDictionary](#), [ActiveHyphenationDictionary](#), [ActiveSpellingDictionary](#), and [ActiveThesaurusDictionary](#) properties to return the corresponding **Dictionary** object. The following example returns the full path for the active spelling dictionary used in the U.S. English version of Word.

```
Sub ShowDictionaryPath
    Set myspell = Languages(wdEnglishUS).ActiveSpellingDictionary
    MsgBox myspell.Path & Application.PathSeparator & myspell.Name
End Sub
```

Setting the Writing Style

The writing style is the set of rules used by the grammar checker. The [WritingStyleList](#) property returns an array of strings that represent the available writing styles for the specified language. The following example returns the list of writing styles for U.S. English.

```
Sub ListWritingStyles()  
    WrStyles = Languages(wdEnglishUS).WritingStyleList  
    For i = 1 To UBound(WrStyles)  
        MsgBox WrStyles(i)  
    Next i  
End Sub
```

Use the [DefaultWritingStyle](#) property to set the default writing style you want Word to use.

```
Languages(wdEnglishUS).DefaultWritingStyle = "Casual"
```

You can override the default writing style with the [ActiveWritingStyle](#) property. This property is applied to a specified document for text marked in a specified language. The following example sets the writing style to be used for checking U.S. English, French, and German in the active document.

```
Sub SetWritingStyle()  
    With ActiveDocument  
        .ActiveWritingStyle(wdEnglishUS) = "Technical"  
        .ActiveWritingStyle(wdFrench) = "Commercial"  
        .ActiveWritingStyle(wdGerman) = "Technisch/Wiss"  
    End With  
End Sub
```

Remarks

You must have the proofing tools installed for each language you intend to check. For more information on working in other languages, see [Language-specific information](#).

If you mark text as **wdNoProofing**, Word skips the marked text when running a spelling or grammar check.

Languages Collection Object

[Application](#) | [Languages \(Language\)](#)
| [Dictionaries \(Dictionary\)](#)

A collection of **Language** objects that represent languages used for proofing or formatting in Word.

Using the Languages Collection

Use the **Languages** property to return the **Languages** collection. The following example displays the localized name for each language.

```
For Each la In Languages
    MsgBox la.NameLocal
Next la
```

Use **Languages(index)** to return a single **Language** object, where *index* can be the value of the **Name** property, the value of the **NameLocal** property, one of the **WdLanguageID** constants, or one of the **MsoLanguageID** constants. (For the list of valid **WdLanguageID** or **MsoLanguageID** constants, see the Object Browser in the Visual Basic Editor.)

Remarks

The **Count** property returns the number of languages for which you can mark text (languages for which proofing tools are available). To check proofing, you must install the appropriate tools for each language you intend to check. You need both a .dll file and an .lex file for each of the following: the thesaurus, spelling checker, grammar checker, and hyphenation tools.

If you mark text as **wdNoProofing**, Word skips the marked text when running a spelling or grammar check. To mark text for a specified language or for no proofing, use the **Set Language** command (**Tools** menu, **Language** sub menu).

LetterContent Object

[Documents \(Document\)](#)  [LetterContent](#)

Represents the elements of a letter created by the Letter Wizard.

Using the LetterContent Object

Use the [GetLetterContent](#) method or the [CreateLetterContent](#) method to return a **LetterContent** object. The following example retrieves and displays the letter recipient's name from the active document.

```
Set myLetterContent = ActiveDocument.GetLetterContent  
MsgBox myLetterContent.RecipientName
```

The following example uses the **CreateLetterContent** method to create a new **LetterContent** object, which is then used with the **RunLetterWizard** method.

```
Set myLetter = ActiveDocument _  
    .CreateLetterContent(DateFormat:="July 11, 1996", _  
    IncludeHeaderFooter:=False, _  
    PageDesign:="C:\MSOffice\Templates\Letters & " _  
        & "Faxes\Contemporary Letter.dot", _  
    LetterStyle:=wdFullBlock, Letterhead:=True, _  
    LetterheadLocation:=wdLetterTop, _  
    LetterheadSize:=InchesToPoints(1.5), _  
    RecipientName:"Dave Edson", _  
    RecipientAddress:"100 Main St." & vbCrLf _  
        & "Bellevue, WA 98004", _  
    Salutation:"Dear Dave,", _  
    SalutationType:=wdSalutationInformal, _  
    RecipientReference:"", MailingInstructions:"", _  
    AttentionLine:"", _  
    Subject:"End of year report", CCList:"", ReturnAddress:"", _  
    SenderName:"", Closing:"Sincerely yours,", _  
    SenderCompany:"", _  
    SenderJobTitle:"", SenderInitials:"", EnclosureNumber:=0)  
ActiveDocument.RunLetterWizard _  
    LetterContent:=myLetter, WizardMode:=True
```

Remarks

The **CreateLetterContent** method creates a **LetterContent** object; however, there are numerous required arguments. If you want to set only a few properties, use the **New** keyword to create a new, stand-alone **LetterContent** object. The following example creates a **LetterContent** object, sets some of its properties, and then uses the **LetterContent** object with the [RunLetterWizard](#) method to run the Letter Wizard, using the preset values as the default settings.

```
Set myLetter = New LetterContent
With myLetter
    .AttentionLine = "Read this"
    .EnclosureNumber = 1
    .Letterhead = True
    .LetterheadLocation = wdLetterTop
    .LetterheadSize = InchesToPoints(2)
End With
Documents.Add.RunLetterWizard LetterContent:=myLetter, _
    WizardMode:=True
```

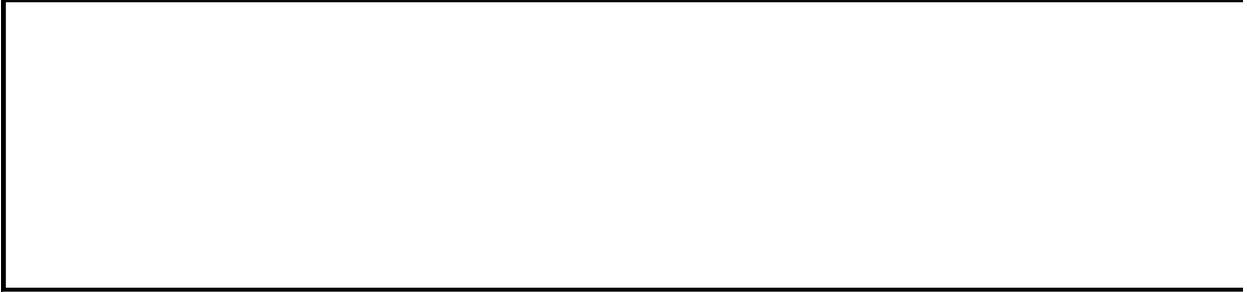
You can duplicate a **LetterContent** object by using the [Duplicate](#) property. The following example retrieves the letter elements in the active document and makes a duplicate copy. The example assigns the duplicate copy to `aLetter` and resets the recipient's name and address to empty strings. The **RunLetterWizard** method is used to run the Letter Wizard, using the values in the revised **LetterContent** object (`aLetter`) as the default settings.

```
Set aLetter = ActiveDocument.GetLetterContent.Duplicate
With aLetter
    .RecipientName = ""
    .RecipientAddress = ""
End With
Documents.Add.RunLetterWizard LetterContent:=aLetter, _
    WizardMode:=True
```

The [SetLetterContent](#) method inserts the contents of the specified **LetterContent** object in a document. The following example retrieves the letter elements from the active document, changes the attention line, and then uses the **SetLetterContent** method to update the active document to reflect the change.

```
Set myLetterContent = ActiveDocument.GetLetterContent
```

```
myLetterContent.AttentionLine = "Greetings"  
ActiveDocument.SetLetterContent LetterContent:=myLetterContent
```



LineFormat Object

[Shapes \(Shape\)](#)  [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.

```
With ActiveDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```



LineNumbering Object

[PageSetup](#) └ [LineNumbering](#)

Represents line numbers in the left margin or to the left of each newspaper-style column.

Using the LineNumbering Object

Use the [LineNumbering](#) property to return the **LineNumbering** object. The following example applies line numbering to the text in the first section of the active document.

```
With ActiveDocument.Sections(1).PageSetup.LineNumbering
    .Active = True
    .CountBy = 5
    .RestartMode = wdRestartPage
End With
```

The following example applies line numbering to the pages in the current section.

```
Selection.PageSetup.LineNumbering.Active = True
```



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, inline shape, or field to return the **LinkFormat** object. The following example breaks the link for the first shape on the active document.

```
ActiveDocument.Shapes(1).LinkFormat.BreakLink
```

Remarks

Not all types of shapes, inline shapes, and fields can be linked to a source. Use the **Type** property for the **Shape** and **InlineShape** objects to determine whether a particular shape can be linked. The **Type** property for a **Field** object returns the type of field.

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



List Object

Multiple objects [└ Lists \(List\)](#)
└ Multiple objects

Represents a single list format that's been applied to specified paragraphs in a document. The **List** object is a member of the [Lists](#) collection.

Using the List Object

Use **Lists**(*index*), where *index* is the index number, to return a single **List** object. The following example returns the number of items in list one in the active document.

```
mycount = ActiveDocument.Lists(1).CountNumberedItems
```

To return all the paragraphs that have list formatting, use the [ListParagraphs](#) property. To return them as a range, use the [Range](#) property.

Remarks

To apply a different list format to an existing list, use the [ApplyListTemplate](#) method with the **List** object. To add a new list to a document, use the **ApplyListTemplate** method with the **ListFormat** object for a specified range.

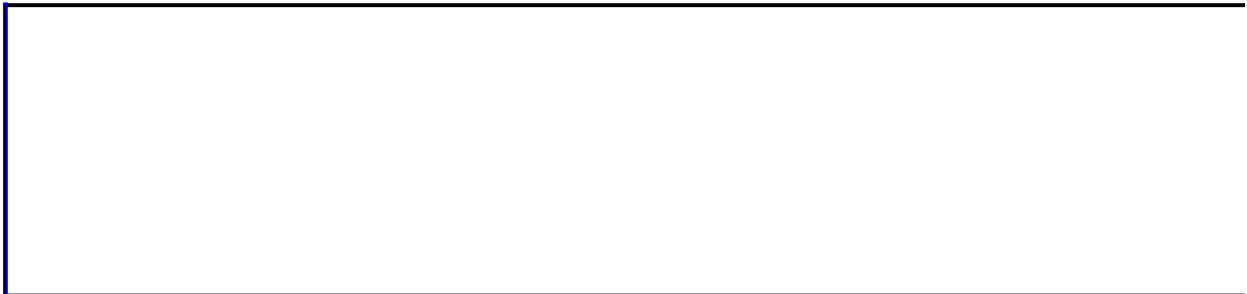
Use the [CanContinuePreviousList](#) method to determine whether you can continue the list formatting from a list that was previously applied to the document.

Use the [CountNumberedItems](#) method to return the number of items in a numbered or bulleted list, including LISTNUM fields.

To determine whether a list contains more than one list template, use the [SingleListTemplate](#) property.

You can manipulate the individual **List** objects within a document, but for more precise control you should work with the [ListFormat](#) object.

Picture-bulleted lists are not included in the **Lists** collection and cannot be manipulated using the **List** object.



ListEntries Collection Object

[FormFields \(FormField\)](#) [└ DropDown](#)
[└ ListEntries \(ListEntry\)](#)

A collection of [ListEntry](#) objects that represent all the items in a drop-down form field.

Using the ListEntries Collection

Use the **ListEntries** property to return the **ListEntries** collection. The following example displays the items that appear in the form field named "Drop1."

```
For Each le In _
    ActiveDocument.FormFields("Drop1").DropDown.ListEntries
    MsgBox le.Name
Next le
```

Use the [Add](#) method to add an item to a drop-down form field. The following example inserts a drop-down form field and then adds "red," "blue," and "green" to the form field.

```
Set myField = _
    ActiveDocument.FormFields.Add(Range:=Selection.Range, _
        Type:=wdFieldFormDropDown)
With myField.DropDown.ListEntries
    .Add Name:="Red"
    .Add Name:="Blue"
    .Add Name:="Green"
End With
```

Use **ListEntries(index)**, where *index* is the list entry name or the index number, to return a single **ListEntry** object. The index number represents the position of the entry in the drop-down form field (the first item is index number 1). The following example deletes the "Blue" entry from the drop-down form field named "Color."

```
ActiveDocument.FormFields("Color").DropDown _
    .ListEntries("Blue").Delete
```

The following example displays the first item in the drop-down form field named "Color."

```
MsgBox _
    ActiveDocument.FormFields("Color").DropDown.ListEntries(1).Name
```



ListEntry Object

[FormFields \(FormField\)](#) [└ DropDown](#)
[└ ListEntries \(ListEntry\)](#)

Represents an item in a drop-down form field. The **ListEntry** object is a member of the [ListEntries](#) collection. The **ListEntries** collection includes all the items in a drop-down form field.

Using the ListEntry Object

Use **ListEntries**(*index*), where *index* is the list entry name or the index number, to return a single **ListEntry** object. The index number represents the position of the entry in the drop-down form field (the first item is index number 1). The following example deletes the "Blue" entry from the drop-down form field named "Color."

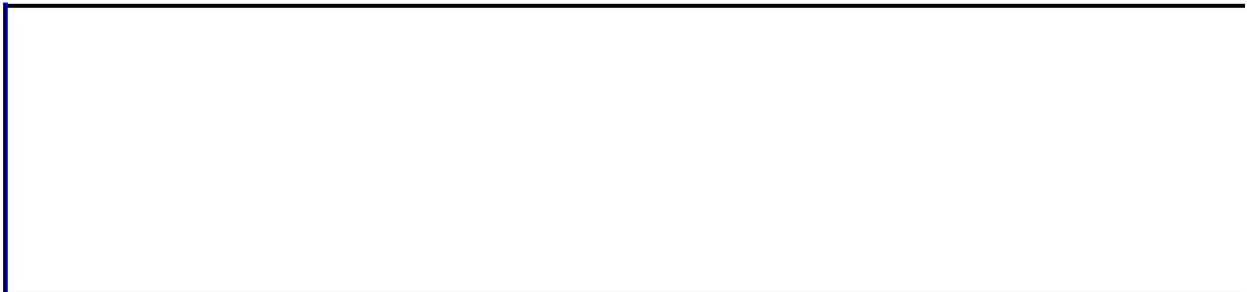
```
ActiveDocument.FormFields("Color").DropDown _  
    .ListEntries("Blue").Delete
```

The following example displays the first item in the drop-down form field named "Color."

```
MsgBox _  
    ActiveDocument.FormFields("Color").DropDown.ListEntries(1).Name
```

Use the [Add](#) method to add an item to a drop-down form field. The following example inserts a drop-down form field and then adds "red," "blue," and "green" to the form field.

```
Set myField = _  
    ActiveDocument.FormFields.Add(Range:=Selection.Range, _  
    Type:=wdFieldFormDropDown)  
With myField.Dropdown.ListEntries  
    .Add Name:="Red"  
    .Add Name:="Blue"  
    .Add Name:="Green"  
End With
```



ListFormat Object

[Range](#)  [ListFormat](#)
 Multiple objects

Represents the list formatting attributes that can be applied to the paragraphs in a range.

Using the ListFormat Object

Use the **ListFormat** property to return the **ListFormat** object for a range. The following example applies the default bulleted list format to the selection.

```
Selection.Range.ListFormat.ApplyBulletDefault
```

Applying a List Template

An easy way to apply list formatting is to use the [ApplyBulletDefault](#), [ApplyNumberDefault](#), and [ApplyOutlineNumberDefault](#) methods, which correspond, respectively, to the first list format (excluding **None**) on each tab in the **Bullets and Numbering** dialog box.

To apply a format other than the default format, use the [ApplyListTemplate](#) method, which allows you to specify the list format (list template) you want to apply.

Returning the List or List Template

Use the [List](#) or [ListTemplate](#) property to return the list or list template from the first paragraph in the specified range.

Remarks

Use the **ListFormat** property with a **Range** object to access the list formatting properties and methods available for the specified range. The following example applies the default bullet list format to the second paragraph in the active document.

```
ActiveDocument.Paragraphs(2).Range.ListFormat.ApplyBulletDefault
```

However, if there's already a list defined in your document, you can access a **List** object by using the [Lists](#) property. The following example changes the format of the list created in the preceding example to the first number format on the **Numbered** tab in the **Bullets and Numbering** dialog box.

```
ActiveDocument.Lists(1).ApplyListTemplate _  
    ListTemplate:=ListGalleries(2).ListTemplates(1)
```



ListGalleries Collection Object

[Application](#) | [ListGalleries \(ListGallery\)](#)
| [ListTemplates \(ListTemplate\)](#)

A collection of [ListGallery](#) objects that represent the three tabs in the **Bullets and Numbering** dialog box.

Using the ListGalleries Collection

Use the **ListGalleries** property to return the **ListGalleries** collection. The following example enumerates the collection of list galleries and sets each of the seven list templates (formats) back to the list template format built into Word.

```
For Each lg In ListGalleries
    For x = 1 To 7
        lg.Reset(x)
    Next x
Next lg
```

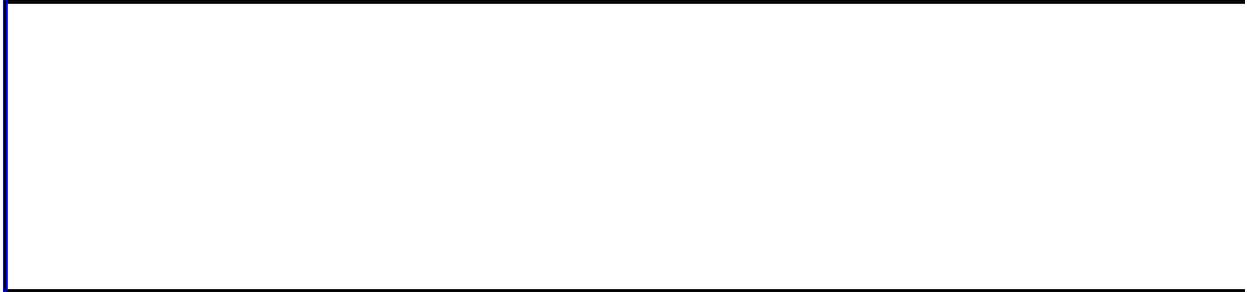
Use **ListGalleries(index)**, where *index* is **wdBulletGallery**, **wdNumberGallery**, or **wdOutlineNumberGallery**, to return a single **ListGallery** object.

The following example returns the third list format (excluding **None**) on the **Bulleted** tab in the **Bullets and Numbering** dialog box and then applies it to the selection.

```
Set temp3 = ListGalleries(wdBulletGallery).ListTemplates(3)
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:= temp3
```

Resetting a List Template in the Gallery

To see whether the specified list template contains the formatting built into Word, use the [Modified](#) property with the **ListGallery** object. To reset formatting to the original list format, use the [Reset](#) method for the **ListGallery** object.



ListGallery Object

[Application](#) | [ListGalleries \(ListGallery\)](#)
| [ListTemplates \(ListTemplate\)](#)

Represents a single gallery of list formats. The **ListGallery** object is a member of the [ListGalleries](#) collection. Each **ListGallery** object represents one of the three tabs in the **Bullets and Numbering** dialog box.

Using the ListGallery Object

Use **ListGalleries**(*index*), where *index* is **wdBulletGallery**, **wdNumberGallery**, or **wdOutlineNumberGallery**, to return a single **ListGallery** object.

The following example returns the third list format (excluding **None**) on the **Bulleted** tab in the **Bullets and Numbering** dialog box and then applies it to the selection.

```
Set temp3 = ListGalleries(wdBulletGallery).ListTemplates(3)
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:= temp3
```

Resetting a List Template in the Gallery

To see whether the specified list template contains the formatting built into Word, use the [Modified](#) property for the **ListGallery** object. To reset formatting to the original list format, use the [Reset](#) method for the **ListGallery** object.



ListLevel Object

[ListLevels](#) └ [ListLevel](#)
└ Multiple objects

Represents a single list level, either the only level for a bulleted or numbered list or one of the nine levels of an outline numbered list. The **ListLevel** object is a member of the [ListLevels](#) collection.

Using the ListLevel Object

Use **ListLevels**(*index*), where *index* is a number from 1 through 9, to return a single **ListLevel** object. The following example sets list level one of list template one in the active document to start at 4.

```
ActiveDocument.ListTemplates(1).ListLevels(1).StartAt = 4
```

Remarks

The **ListLevel** object gives you access to all the formatting properties for the specified list level, such as the [Alignment](#), [Font](#), [NumberFormat](#), [NumberPosition](#), [NumberStyle](#), and [TrailingCharacter](#) properties.

To apply a list level, first identify the range or list, and then use the [ApplyListTemplate](#) method. Each tab at the beginning of the paragraph is translated into a list level. For example, a paragraph that begins with three tabs will become a level-three list paragraph after the **ApplyListTemplate** method is used.



ListLevels Collection Object

[ListTemplate](#) └ [ListLevels](#)
└ [ListLevel](#)

A collection of [ListLevel](#) objects that represents all the list levels of a list template, either the only level for a bulleted or numbered list or one of the nine levels of an outline numbered list.

Using the ListLevels Collection

Use the [ListLevels](#) property to return the **ListLevels** collection. The following example sets the variable `mytemp` to the first list template in the active document and then modifies each level to use lowercase letters for its number style.

```
Set mytemp = ActiveDocument.ListTemplates(1)
For Each lev In mytemp.ListLevels
    lev.NumberStyle = wdListNumberStyleLowercaseLetter
Next lev
```

Use **ListLevels**(*index*), where *index* is a number from 1 through 9, to return a single **ListLevel** object. The following example sets list level one of list template one in the active document to start at four.

```
ActiveDocument.ListTemplates(1).ListLevels(1).StartAt = 4
```

Note You cannot add new levels to a list template.

Remarks

To apply a list level, first identify the range or list, and then use the [ApplyListTemplate](#) method. Each tab at the beginning of the paragraph is translated into a list level. For example, a paragraph that begins with three tabs will become a level-three list paragraph after the **ApplyListTemplate** method is used.

ListParagraphs Collection Object

Multiple objects [└ ListParagraphs](#)
[└ Paragraph](#)

A collection of [Paragraph](#) objects that represents the paragraphs of the specified document, list, or range that have list formatting applied.

Using the ListParagraphs Collection

Use the [ListParagraphs](#) property to return the **ListParagraphs** collection. The following example applies highlighting to the collection of paragraphs with list formatting in the active document.

```
For Each para in ActiveDocument.ListParagraphs
    para.Range.HighlightColorIndex = wdTurquoise
Next para
```

Use **ListParagraphs(index)**, where *index* is the index number, to return a single [Paragraph](#) object with list formatting.

Remarks

Paragraphs can have two types of list formatting. The first type includes an automatically added number or bullet at the beginning of each paragraph in the list. The second type includes LISTNUM fields, which can be placed anywhere inside a paragraph. There can be more than one LISTNUM field per paragraph.

To add list formatting to paragraphs, you can use the [ApplyListTemplate](#), [ApplyBulletDefault](#), [ApplyNumberDefault](#), or [ApplyOutlineNumberDefault](#) method. You access these methods through the **ListFormat** object for a specified range.

The [Count](#) property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the [Range](#) object.



Lists Collection Object

Multiple objects [└ Lists \(List\)](#)
└ Multiple objects

A collection of [List](#) objects that represent all the lists in the specified document.

Using the Lists Collection

Use the **Lists** property to return the **Lists** collection. The following example displays the number of items in each list in the active document.

```
For Each li In ActiveDocument.Lists
    MsgBox li.CountNumberedItems
Next li
```

Use **Lists(index)**, where *index* is the index number, to return a single **List** object. The following example applies the first list format (excluding **None**) on the **Numbered** tab in the **Bullets and Numbering** dialog box to the second list in the active document.

```
Set temp1 = ListGalleries(wdNumberGallery).ListTemplates(1)
ActiveDocument.Lists(2).ApplyListTemplate ListTemplate:=temp1
```

Remarks

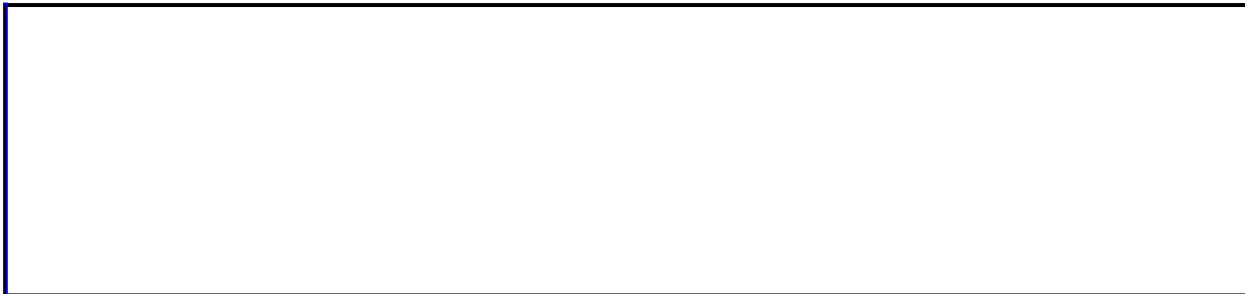
When you use a **For Each...Next** loop to enumerate the **Lists** collection, the lists in a document are returned in reverse order. The following example counts the items for each list in the active document, from the bottom of the document upward.

```
For Each li In ActiveDocument.Lists
    MsgBox li.CountNumberedItems
Next li
```

To add a new list to a document, use the [ApplyListTemplate](#) method with the **ListFormat** object for a specified range.

You can manipulate the individual **List** objects within a document, but for more precise control you should work with the [ListFormat](#) object.

Picture-bulleted lists are not included in the **Lists** collection.



ListTemplate Object

Multiple objects [└ ListTemplates \(ListTemplate\)](#)
[└ ListLevels \(ListLevel\)](#)

Represents a single list template that includes all the formatting that defines a list. The **ListTemplate** object is a member of the [ListTemplates](#) collection. Each of the seven formats (excluding **None**) found on each of the three tabs in the **Bullets and Numbering** dialog box corresponds to a list template object. These predefined list templates can be accessed from the three **ListGallery** objects in the **ListGalleries** collection. Documents and templates can also contain collections of list templates.

Using the ListTemplate Object

Use **ListTemplates**(*index*), where *index* is a number from 1 through 7, to return a single list template from a list gallery. The following example returns the third list format (excluding **None**) on the **Numbered** tab in the **Bullets and Numbering** dialog box.

```
Set temp3 = ListGalleries(2).ListTemplates(3)
```

Note Some properties and methods — **Convert** and **Add**, for example — won't work with list templates that are accessed from a list gallery. You can modify these list templates, but you cannot change their list gallery type (**wdBulletGallery**, **wdNumberGallery**, or **wdOutlineNumberGallery**).

The following example sets an object variable equal to the list template used in the third list in the active document, and then it applies that list template to the selection.

```
Set myLt = ActiveDocument.ListTemplates(3)  
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=myLt
```

Use the [Add](#) method to add a list template to the collection of list templates in a document or template.

Resetting a List Template in the Gallery

To see whether the specified list template contains the formatting built into Word, use the [Modified](#) property with the **ListGallery** object. To reset formatting to the original list format, use the [Reset](#) method for the **ListGallery** object.

Remarks

After you have returned a **ListTemplate** object, use **ListLevels(*index*)**, where *index* is a number from 1 through 9, to return a single **ListLevel** object. With a **ListLevel** object, you have access to all the formatting properties for the specified list level, such as **Alignment**, **Font**, **NumberFormat**, **NumberPosition**, **NumberStyle**, and **TrailingCharacter**.

Use the [Convert](#) method to convert a multiple-level list template to a single-level template.



ListTemplates Collection Object

Multiple objects [└ ListTemplates \(ListTemplate\)](#)
[└ ListLevels \(ListLevel\)](#)

A collection of [ListTemplate](#) objects that represent the seven predefined list formats on each tab in the **Bullets and Numbering** dialog box.

Using the ListTemplates Collection

Use the **ListTemplates** property to return the **ListTemplates** collection. The following example displays a message with the level status (single or multiple-level) for each list template in the active document.

```
For Each lt In ActiveDocument.ListTemplates
    MsgBox "This is a multiple-level list template - " &
        & lt.OutlineNumbered
Next LT
```

Use the [Add](#) method to add a list template to the collection in the specified document or template. The following example adds a new list template to the active document and applies it to the selection.

```
Set myLT = ActiveDocument.ListTemplates.Add
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=myLT
```

Use **ListTemplates(index)**, where *index* is a number 1 through 7, to return a single list template from a list gallery. The following example sets an object variable equal to the list template used in the third list in the active document, and then it applies that list template to the selection.

```
Set mylt = ActiveDocument.ListTemplates(3)
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=mylt
```

Note Some properties and methods — **Convert** and **Add**, for example — won't work with list templates that are accessed from a list gallery. You can modify these list templates, but you cannot change their list gallery type (**wdBulletGallery**, **wdNumberGallery**, or **wdOutlineNumberGallery**).

Resetting a List Template in the Gallery

To see whether the specified list template contains the formatting built into Word, use the [Modified](#) property with the **ListGallery** object. To reset formatting to the original list format, use the [Reset](#) method for the **ListGallery** object.

Remarks

After you have returned a **ListTemplate** object, use **ListLevels(*index*)**, where *index* is a number from 1 through 9, to return a single **ListLevel** object. With a **ListLevel** object, you have access to all the formatting properties for the specified list level, such as **Alignment**, **Font**, **NumberFormat**, **NumberPosition**, **NumberStyle**, and **TrailingCharacter**.

Use the [Convert](#) method to convert a multiple-level list template to a single-level template.



MailingLabel Object

[Application](#) └ [MailingLabel](#)
└ Multiple objects

Represents a mailing label.

Using the MailingLabel Object

Use the [MailingLabel](#) property to return the **MailingLabel** object. The following example sets default mailing label options.

```
With Application.MailingLabel
    .DefaultLaserTray = wdPrinterLowerBin
    .DefaultPrintBarCode = True
End With
```

Use the [PrintOut](#) method to print a mailing label listed in the **Product Number** box in the **Label Options** dialog box. The following example prints a page of Avery 5162 standard address labels using the specified address.

```
addr = "Katie Jordan" & vbCr & "123 Skye St." _
    & vbCr & "OurTown, WA 98107"
Application.MailingLabel.PrintOut Name:="5162", Address:=addr
```

Remarks

Use the [CustomLabels](#) property to format or print a custom mailing label. The following example sets the number of labels across and down for the custom label named "MyLabel."

```
With Application.MailingLabel.CustomLabels("MyLabel")  
    .NumberAcross = 2  
    .NumberDown = 5  
End With
```



MailMerge Object

[Document](#) └ [MailMerge](#)
└ Multiple objects

Represents the mail merge functionality in Word.

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 operation has begun. Use the [State](#) property to determine the status of the mail merge operation. The following example executes a mail merge if the active document is a main document with an attached data source.

```
If ActiveDocument.MailMerge.State = wdMainAndDataSource Then
    ActiveDocument.MailMerge.Execute
End If
```

The following example merges the main document with the first three data records in the attached data source and then sends the results to the printer.

```
Set myMerge = ActiveDocument.MailMerge
If myMerge.State = wdMainAndSourceAndHeader Or _
    myMerge.State = wdMainAndDataSource Then
    With myMerge.DataSource
        .FirstRecord = 1
        .LastRecord = 3
    End With
End If
With myMerge
    .Destination = wdSendToPrinter
    .Execute
End With
```



MailMergeDataField Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeDataSource](#)
└ [MailMergeDataFields \(MailMergeDataField\)](#)

Represents a single mail 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 data source (for example, Name, Address, and City).

Using the MailMergeDataField Object

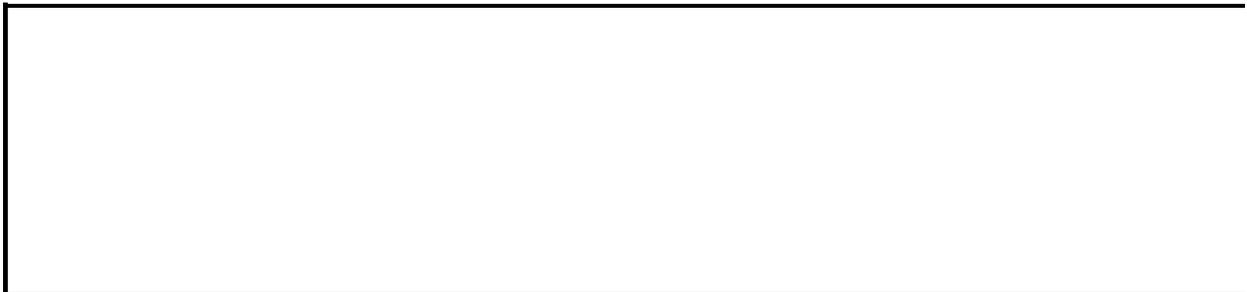
Use **DataFields**(*index*), where *index* is the data field name or the index number, to return a single **MailMergeDataField** object. The index number represents the position of the data field in the mail merge data source. The following example retrieves the first value from the FName field in the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

```
first = _  
    ActiveDocument.MailMerge.DataSource.DataFields("FName").Value
```

The following example displays the name of first field in the data source attached to the active document.

```
MsgBox ActiveDocument.MailMerge.DataSource.DataFields(1).Name
```

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



MailMergeDataFields Collection Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeDataSource](#)
└ [MailMergeDataFields \(MailMergeDataField\)](#)

A collection of [MailMergeDataField](#) objects that represent the data fields in a mail merge data source.

Using the MailMergeDataFields Collection

Use the **DataFields** property to return the **MailMergeDataFields** collection. The following example displays the names of all the fields in the attached data source.

```
For Each afield In ActiveDocument.MailMerge.DataSource.DataFields
    MsgBox afield.Name
Next afield
```

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. Use the [EditDataSource](#) method to edit the contents of a data source. The following example adds a data field named "Author" to a table in the attached data source.

```
If ActiveDocument.MailMerge.DataSource.Type = _
    wdMergeInfoFromWord Then
    ActiveDocument.MailMerge.EditDataSource
    With ActiveDocument.Tables(1)
        .Columns.Add
        .Cell(Row:=1, Column:=.Columns.Count).Range.Text = "Author"
    End With
End If
```

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. The following example retrieves the first value from the FName field in the data source attached to the active document.

```
first = _
    ActiveDocument.MailMerge.DataSource.DataFields("FName").Value
```

The following example displays the name of first data field in the data source attached to the active document.

```
MsgBox ActiveDocument.MailMerge.DataSource.DataFields(1).Name
```



MailMergeDataSource Object

[MailMerge](#) └ [MailMergeDataSource](#)
└ Multiple objects

Represents the mail merge data source in a mail 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 document.

```
If ActiveDocument.MailMerge.DataSource.Name <> "" Then _  
    MsgBox ActiveDocument.MailMerge.DataSource.Name
```

The following example displays the field names in the data source associated with the active document.

```
For Each aField In ActiveDocument.MailMerge.DataSource.FieldNames  
    MsgBox aField.Name  
Next aField
```

The following example opens the data source associated with Form letter.doc and determines whether the FirstName field includes the name "Kate."

```
With Documents("Form letter.doc").MailMerge  
    .EditDataSource  
    If .DataSource.FindRecord(FindText:="Kate", _  
        Field:="FirstName") = True Then  
        MsgBox "Data was found"  
    End If  
End With
```



MailMergeField Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeFields \(MailMergeField\)](#)
└ [Range](#)

Represents a single mail merge field in a document. The **MailMergeDataField** object is a member of the [MailMergeDataFields](#) collection. The **MailMergeDataFields** collection includes all the mail merge related fields in a document.

Using the MailMergeField Object

Use **Fields**(*index*), where *index* is the index number, to return a single **MailMergeField** object. The following example displays the field code of the first mail merge field in the active document.

```
MsgBox ActiveDocument.MailMerge.Fields(1).Code
```

Use the [Add](#) method to add a merge field to the **MailMergeFields** collection. The following example replaces the selection with a MiddleInitial merge field.

```
ActiveDocument.MailMerge.Fields.Add Range:=Selection.Range, _  
    Name:="MiddleInitial"
```

Remarks

The **MailMergeFields** collection has additional methods, such as [AddAsk](#) and [AddFillIn](#), for adding fields related to a mail merge operation.

MailMergeFieldName Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeDataSource](#)
└ [MailMergeFieldNames \(MailMergeFieldName\)](#)

Represents a mail merge field name in a data source. The **MailMergeFieldName** object is a member of the [MailMergeFieldNames](#) collection. The **MailMergeFieldNames** collection includes all the data field names in a mail merge data source.

Using the MailMergeFieldName Object

Use **FieldNames**(*index*), where *index* is the index number, to return a single **MailMergeFieldName** object. The index number represents the position of the field in the mail merge data source. The following example retrieves the name of the last field in the data source attached to the active document.

```
alast = ActiveDocument.MailMerge.DataSource.FieldNames.Count  
afirst = ActiveDocument.MailMerge.DataSource.FieldNames(alast).Name  
MsgBox afirst
```

You cannot add fields to the **MailMergeFieldNames** collection. Field names in a data source are automatically included in the **MailMergeFieldNames** collection.



MailMergeFieldNames Collection Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeDataSource](#)
└ [MailMergeFieldNames \(MailMergeFieldName\)](#)

A collection of [MailMergeFieldName](#) objects that represent the field names in a mail merge data source.

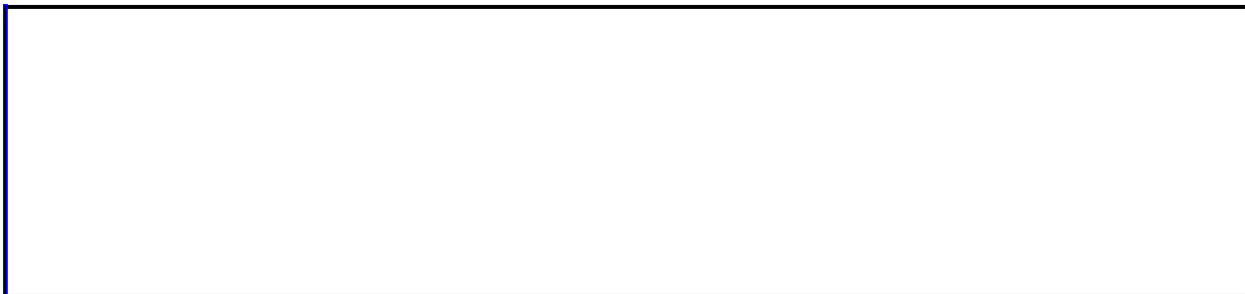
Using the MailMergeFieldNames Collection

Use the **FieldNames** property to return the **MailMergeFieldNames** collection. The following example displays the names of the fields in the data source attached to the active document.

```
For Each afield In ActiveDocument.MailMerge.DataSource.FieldNames
    MsgBox afield.Name
Next afield
```

You cannot add names to the **MailMergeFieldNames** collection. When a field is added to a data source, the field name is automatically included in the **MailMergeFieldNames** collection. Use the [EditDataSource](#) method to edit the contents of a data source. The following example adds a data field named "Author" to a table in the data source attached to the active document.

```
If ActiveDocument.MailMerge.DataSource.Type = _
    wdMergeInfoFromWord Then
    ActiveDocument.MailMerge.EditDataSource
    With ActiveDocument.Tables(1)
        .Columns.Add
        .Cell(Row:=1, Column:=.Columns.Count).Range.Text = "Author"
    End With
End If
```



MailMergeFields Collection Object

[Documents \(Document\)](#) └ [MailMerge](#)
└ [MailMergeFields \(MailMergeField\)](#)
└ [Range](#)

A collection of **[MailMergeField](#)** objects that represent the mail merge related fields in a document.

Using the MailMergeFields Collection

Use the **Fields** property to return the **MailMergeFields** collection. The following example adds an ASK field after the last mail merge field in the active document.

```
Set myMMFields = ActiveDocument.MailMerge.Fields
myMMFields(myMMFields.Count).Select
Selection.MoveRight Unit:=wdWord, Count:=1, Extend:=wdMove
ActiveDocument.MailMerge.Fields.AddAsk Range:=Selection.Range, _
    Name:="Name", Prompt:="Type your name", AskOnce:=True
```

Use the **Add** method to add a merge field to the **MailMergeFields** collection. The following example replaces the selection with a **MiddleInitial** merge field.

```
ActiveDocument.MailMerge.Fields.Add Range:=Selection.Range, _
    Name:="MiddleInitial"
```

Use **Fields(index)**, where *index* is the index number, to return a single **MailMergeField** object. The following example displays the field code of the first mail merge field in the active document.

```
MsgBox ActiveDocument.MailMerge.Fields(1).Code
```

Remarks

The **MailMergeFields** collection has additional methods, such as [AddAsk](#) and [AddFillIn](#), for adding fields related to a mail merge operation.

MailMessage Object

[Application](#)  [MailMessage](#)

Represents the active email message if you are using Word as your e-mail editor.

Using the MailMessage Object

Use the **MailMessage** property to return the **MailMessage** object. The following example validates the e-mail addresses that appear in the active e-mail message.

```
Application.MailMessage.CheckName
```

Remarks

The methods of the **MailMessage** object require that you are using Word as your e-mail editor and that an e-mail message is active. If either of these conditions isn't true, an error occurs.

MappedDataField Object

[MappedDataFields](#)  [MappedDataField](#)

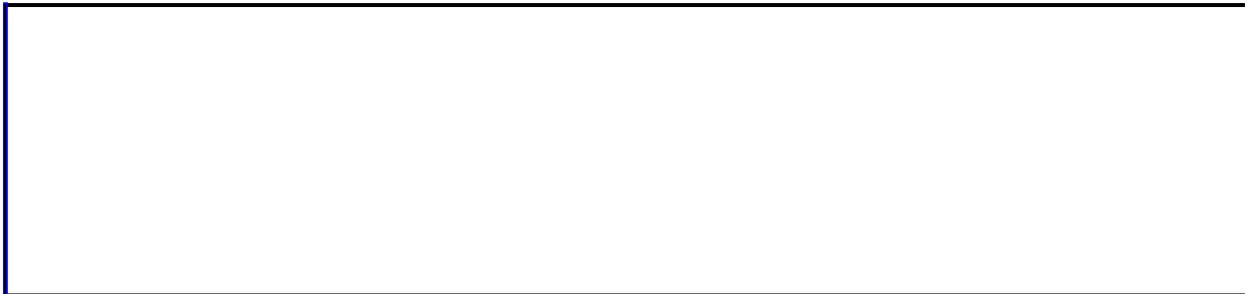
Represents a single mapped data field. The **MappedDataField** object is a member of the [MappedDataFields](#) collection. The **MappedDataFields** collection includes all the mapped data fields available in Microsoft Word.

A mapped data field is a field contained within Microsoft Word 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 in Word. If a document or template is to be merged with more than one data source, mapped data fields make it unnecessary to reenter the fields into the document to agree with the field names in the database.

Using the MappedDataField object

Use the [MappedDataFields](#) property to return a **MappedDataField** object. This example returns the data source field name for the **wdFirstName** mapped data field. This example assumes the current document is a mail merge document. 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()  
  
    With ThisDocument.MailMerge.DataSource  
        If .MappedDataFields.Item(wdFirstName).DataFieldName <> "" T  
            MsgBox "The mapped data field 'FirstName' is mapped to "  
                & .MappedDataFields(Index:=wdFirstName) _  
                .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
```



↳ [Show All](#)

MappedDataFields Collection

[MailMergeDataSource](#)  [MappedDataFields](#)
 [MappedDataField](#)

A collection of **MappedDataField** objects that represents all the [mapped data fields](#) available in Microsoft Word.

Using the MappedDataFields collection

Use the [MappedDataFields](#) property of the [MailMergeDataSource](#) object to return the **MappedDataFields** collection. This example creates a tabbed list of the mapped data fields available in Word and the fields in the data source to which they are mapped. This example assumes that the current document is a mail merge document and that the data source fields have corresponding mapped data fields.

```
Sub MappedFields()  
    Dim intCount As Integer  
    Dim docCurrent As Document  
    Dim docNew As Document  
  
    On Error Resume Next  
  
    Set docCurrent = ThisDocument  
    Set docNew = Documents.Add  
  
    'Add leader tab to new document  
    docNew.Paragraphs.TabStops.Add _  
        Position:=InchesToPoints(3.5), _  
        Leader:=wdTabLeaderDots  
  
    With docCurrent.MailMerge.DataSource  
  
        'Insert heading paragraph for tabbed columns  
        docNew.Content.InsertAfter "Word Mapped Data Field" _  
            & vbTab & "Data Source Field"  
  
        Do  
            intCount = intCount + 1  
  
            'Insert Word mapped data field name and the  
            'corresponding data source field name  
            docNew.Content.InsertAfter .MappedDataFields( _  
                Index:=intCount).Name & vbTab & _  
                .MappedDataFields(Index:=intCount) _  
                .DataFieldName  
  
            'Insert paragraph  
            docNew.Content.InsertParagraphAfter  
        Loop Until intCount = .MappedDataFields.Count  
  
    End With  
End Sub
```

End With

End Sub



OLEFormat Object

Multiple objects [OLEFormat](#)

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

Using the OLEFormat Object

Use the [OLEFormat](#) property for a shape, inline shape, or field to return the **OLEFormat** object. The following example displays the class type for the first shape on the active document.

```
MsgBox ActiveDocument.Shapes(1).OLEFormat.ClassType
```

Remarks

Not all types of shapes, inline shapes, and fields have OLE capabilities. Use the **Type** property for the **Shape** and **InlineShape** objects to determine what category the specified shape or inline shape falls into. The **Type** property for a **Field** object returns the type of field.

You can use the **Activate**, **Edit**, **Open**, and **DoVerb** methods to automate an OLE object.

Use the **Object** property to return an object that represents an ActiveX control or OLE object. With this object, you can use the properties and methods of the container application or the ActiveX control.



Options Object

[Application](#) └ [Options](#)

Represents application and document options in Word. 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 three application options for Word.

```
With Options
    .AllowDragAndDrop = True
    .ConfirmConversions = False
    .MeasurementUnit = wdPoints
End With
```



OtherCorrectionsException Object

[AutoCorrect](#) | [OtherCorrectionExceptions \(OtherCorrectionException\)](#)

Represents a single AutoCorrect exception. The **OtherCorrectionsException** object is a member of the [OtherCorrectionsExceptions](#) collection. The **OtherCorrectionsExceptions** collection includes all words that Microsoft Word won't correct automatically. This list corresponds to the list of AutoCorrect exceptions on the **Other Corrections** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu).

Using the OtherCorrectionsException Object

Use **OtherCorrectionsExceptions**(*index*), where *index* is the AutoCorrect exception name or the index number, to return a single **OtherCorrectionsException** object. The following example deletes "WTop" from the list of AutoCorrect exceptions.

```
AutoCorrect.OtherCorrectionsExceptions("WTop").Delete
```

The index number represents the position of the AutoCorrect exception in the **OtherCorrectionsExceptions** collection. The following example displays the name of the first item in the **OtherCorrectionsExceptions** collection.

```
MsgBox AutoCorrect.OtherCorrectionsExceptions(1).Name
```

If the value of the [OtherCorrectionsAutoAdd](#) property is **True**, words are automatically added to the list of AutoCorrect exceptions. Use the [Add](#) method to add an item to the **OtherCorrectionsExceptions** collection. The following example adds "TipTop" to the list of AutoCorrect exceptions.

```
AutoCorrect.OtherCorrectionsExceptions.Add Name:="TipTop"
```



OtherCorrectionsExceptions Collection Object

[AutoCorrect](#) | [OtherCorrectionExceptions \(OtherCorrectionException\)](#)

A collection of [OtherCorrectionsException](#) objects that represents the list of words that Microsoft Word won't correct automatically. This list corresponds to the list of AutoCorrect exceptions on the **Other Corrections** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu).

Using the OtherCorrectionsExceptions Collection

Use the [OtherCorrectionsExceptions](#) property to return the **OtherCorrectionsExceptions** collection. The following example displays the items in this collection.

```
For Each aCap In AutoCorrect.OtherCorrectionsExceptions
    MsgBox aCap.Name
Next aCap
```

If the value of the [OtherCorrectionsAutoAdd](#) property is **True**, words are automatically added to the list of AutoCorrect exceptions. Use the [Add](#) method to add an item to the **OtherCorrectionsExceptions** collection. The following example adds "TipTop" to the list of AutoCorrect exceptions.

```
AutoCorrect.OtherCorrectionsExceptions.Add Name:="TipTop"
```

Use **OtherCorrectionsExceptions(index)**, where *index* is the name or the index number, to return a single **OtherCorrectionsException** object. The following example deletes "WTop" from the list of AutoCorrect exceptions.

```
AutoCorrect.OtherCorrectionsExceptions("WTop").Delete
```

The index number represents the position of the AutoCorrect exception in the **OtherCorrectionsExceptions** collection. The following example displays the name of the first item in the **OtherCorrectionsExceptions** collection.

```
MsgBox AutoCorrect.OtherCorrectionsExceptions(1).Name
```



PageNumber Object

[Sections \(Section\)](#) └ [HeadersFooters \(HeaderFooter\)](#)
└ [PageNumbers \(PageNumber\)](#)

Represents a page number in a header or footer. The **PageNumber** object is a member of the [PageNumbers](#) collection. The **PageNumbers** collection includes all the page numbers in a single header or footer.

Using the PageNumber Object

Use **PageNumbers**(*index*), where *index* is the index number, to return a single **PageNumber** object. In most cases, a header or footer will contain only one page number, which is index number 1. The following example centers the first page number in the primary header in section one in the active document.

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary) _  
    .PageNumbers(1).Alignment = wdAlignPageNumberCenter
```

Use the **Add** method to add a page number (a PAGE field) to a header or footer. The following example adds a page number to the primary footer in the first section and in any subsequent sections. The page number doesn't appear on the first page.

```
With ActiveDocument.Sections(1)  
    .Footers(wdHeaderFooterPrimary).PageNumbers.Add _  
        PageNumberAlignment:=wdAlignPageNumberLeft, _  
        FirstPage:=False  
End With
```



PageNumbers Collection Object

[Sections \(Section\)](#) [HeadersFooters \(HeaderFooter\)](#)
 [PageNumbers \(PageNumber\)](#)

A collection of [PageNumber](#) objects that represent the page numbers in a single header or footer.

Using the PageNumbers Collection

Use the **PageNumbers** property to return the **PageNumbers** collection. The following example starts page numbering at 3 for the first section in the active document.

```
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary) _  
    .PageNumbers.StartingNumber = 3
```

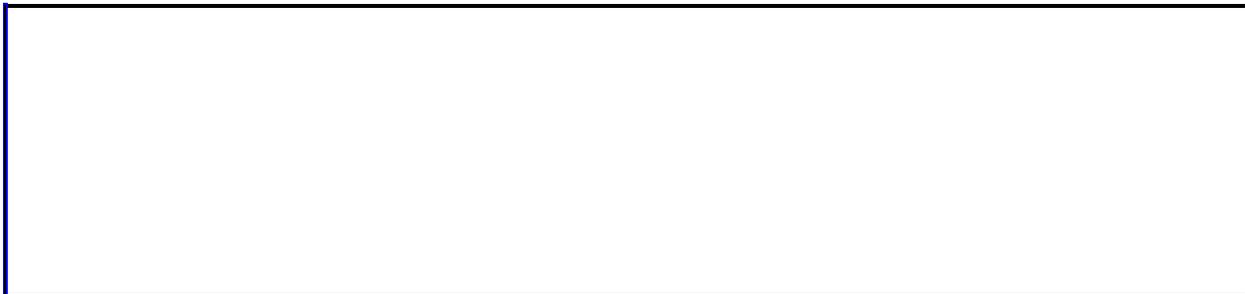
Use the [Add](#) method to add page numbers to a header or footer. The following example adds a page number to the primary footer in the first section.

```
With ActiveDocument.Sections(1)  
    .Footers(wdHeaderFooterPrimary).PageNumbers.Add _  
        PageNumberAlignment:=wdAlignPageNumberLeft, _  
        FirstPage:=False  
End With
```

To add or change page numbers in a document with multiple sections, modify the page numbers in each section or set the [LinkToPrevious](#) property to **True**.

Use **PageNumbers(index)**, where *index* is the index number, to return a single **PageNumber** object. In most cases, a header or footer contains only one page number, which is index number 1. The following example centers the first page number in the primary header in the first section.

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary) _  
    .PageNumbers(1).Alignment = wdAlignPageNumberCenter
```



PageSetup Object

Multiple objects [PageSetup](#)
└ Multiple objects

Represents the page setup description. The **PageSetup** object contains all the page setup attributes of a document (left margin, bottom margin, paper size, and so on) as properties.

Using the PageSetup Object

Use the **PageSetup** property to return the **PageSetup** object. The following example sets the first section in the active document to landscape orientation and then prints the document.

```
ActiveDocument.Sections(1).PageSetup.Orientation = _  
    wdOrientLandscape  
ActiveDocument.PrintOut
```

The following example sets all the margins for the document named "Sales.doc."

```
With Documents("Sales.doc").PageSetup  
    .LeftMargin = InchesToPoints(0.75)  
    .RightMargin = InchesToPoints(0.75)  
    .TopMargin = InchesToPoints(1.5)  
    .BottomMargin = InchesToPoints(1)  
End With
```



Pane Object

[Windows \(Window\)](#) └ [Panels \(Panel\)](#)
└ Multiple objects

Represents a window pane. The **Pane** object is a member of the [Panels](#) collection. The **Panels** collection includes all the window panes for a single window.

Using the Pane Object

Use **Panes**(*index*), where *index* is the index number, to return a single **Pane** object. The following example closes the active pane.

```
If ActiveDocument.ActiveWindow.Panes.Count >= 2 Then _  
    ActiveDocument.ActiveWindow.ActivePane.Close
```

Use the [Add](#) method or the [Split](#) property to add a window pane. The following example splits the active window at 20 percent of the current window size.

```
ActiveDocument.ActiveWindow.Panes.Add SplitVertical:=20
```

The following example splits the active window in half.

```
ActiveDocument.ActiveWindow.Split = True
```

You can use the [SplitSpecial](#) property to show comments, footnotes, or endnotes in a separate pane.

Remarks

A window has more than one pane if the window is split or the view is not print layout view and information such as footnotes or comments are displayed. The following example displays the comments pane in normal view and then prompts to close the pane.

```
ActiveDocument.ActiveWindow.View.Type = wdNormalView
If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.ActiveWindow.View.SplitSpecial = wdPaneComments
    response = _
        MsgBox("Do you want to close the comments pane?", vbYesNo)
    If response = vbYes Then _
        ActiveDocument.ActiveWindow.ActivePane.Close
End If
```



Panes Collection Object

[Windows \(Window\)](#) └ [Panes \(Pane\)](#)
└ Multiple objects

A collection of [Pane](#) objects that represent the window panes for a single window.

Using the Panes Collection

Use the **Panes** property to return the **Panes** collection. The following example splits the active window and hides the ruler for each pane.

```
ActiveDocument.ActiveWindow.Split = True
For Each aPane In ActiveDocument.ActiveWindow.Panes
    aPane.DisplayRulers = False
Next aPane
```

Use the [Add](#) method or the [Split](#) property to add a window pane. The following example splits the active window at 20 percent of the current window size.

```
ActiveDocument.ActiveWindow.Panes.Add SplitVertical:=20
```

The following example splits the active window in half.

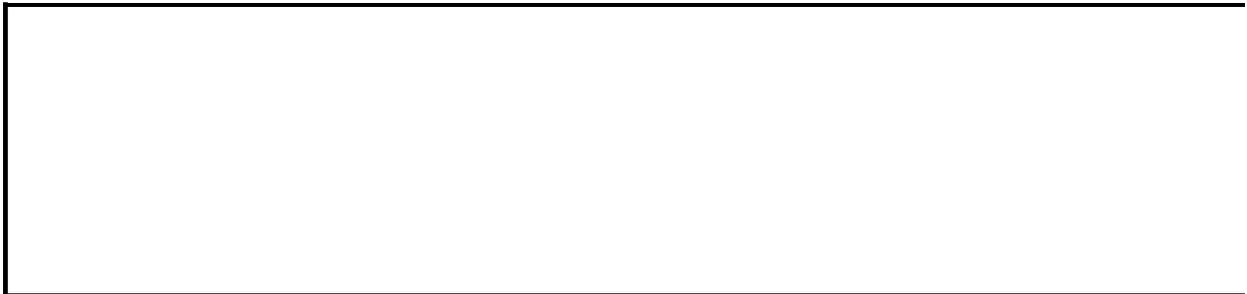
```
ActiveDocument.ActiveWindow.Split = True
```

You can use the [SplitSpecial](#) property to show comments, footnotes, or endnotes in a separate pane.

Remarks

A window has more than one pane if it's split, or if the active view isn't print layout view and information such as footnotes or comments is displayed. The following example displays the footnote pane in normal view and then prompts the user to close the pane.

```
ActiveDocument.ActiveWindow.View.Type = wdNormalView
If ActiveDocument.Footnotes.Count >= 1 Then
    ActiveDocument.ActiveWindow.View.SplitSpecial = wdPaneFootnotes
    response = _
        MsgBox("Do you want to close the footnotes pane?", vbYesNo)
    If response = vbYes Then _
        ActiveDocument.ActiveWindow.ActivePane.Close
End If
```



Paragraph Object

Multiple objects [└ Paragraphs \(Paragraph\)](#)
└ Multiple objects

Represents a single paragraph in a selection, range, or document. The **Paragraph** object is a member of the [Paragraphs](#) collection. The **Paragraphs** collection includes all the paragraphs in a selection, range, or document.

Using the Paragraph Object

Use **Paragraphs**(*index*), where *index* is the index number, to return a single **Paragraph** object. The following example right aligns the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Alignment = wdAlignParagraphRight
```

Use the [Add](#), [InsertParagraph](#), [InsertParagraphAfter](#), or [InsertParagraphBefore](#) method to add a new, blank paragraph to a document. The following example adds a paragraph mark before the first paragraph in the selection.

```
Selection.Paragraphs.Add Range:=Selection.Paragraphs(1).Range
```

The following example also adds a paragraph mark before the first paragraph in the selection.

```
Selection.Paragraphs(1).Range.InsertParagraphBefore
```



ParagraphFormat Object

Multiple objects [↳ ParagraphFormat](#)
↳ Multiple objects

Represents all the formatting for a paragraph.

Using the ParagraphFormat Object

Use the [Format](#) property to return the **ParagraphFormat** object for a paragraph or paragraphs. The [ParagraphFormat](#) property returns the **ParagraphFormat** object for a selection, range, style, [Find](#) object, or [Replacement](#) object. The following example centers the third paragraph in the active document.

```
ActiveDocument.Paragraphs(3).Format.Alignment = _  
    wdAlignParagraphCenter
```

The following example finds the next double-spaced paragraph after the selection.

```
With Selection.Find  
    .ClearFormatting  
    .ParagraphFormat.LineSpacingRule = wdLineSpaceDouble  
    .Text = ""  
    .Forward = True  
    .Wrap = wdFindContinue  
End With  
Selection.Find.Execute
```

Remarks

You can use Visual Basic's **New** keyword to create a new, standalone **ParagraphFormat** object. The following example creates a **ParagraphFormat** object, sets some formatting properties for it, and then applies all of its properties to the first paragraph in the active document.

```
Dim myParaF As New ParagraphFormat
myParaF.Alignment = wdAlignParagraphCenter
myParaF.Borders.Enable = True
ActiveDocument.Paragraphs(1).Format = myParaF
```

You can also make a standalone copy of an existing **ParagraphFormat** object by using the **Duplicate** property. The following example duplicates the paragraph formatting of the first paragraph in the active document and stores the formatting in myDup. The example changes the left indent of myDup to 1 inch, creates a new document, inserts text into the document, and applies the paragraph formatting of myDup to the text.

```
Set myDup = ActiveDocument.Paragraphs(1).Format.Duplicate
myDup.LeftIndent = InchesToPoints(1)
Documents.Add
Selection.InsertAfter "This is a new paragraph."
Selection.Paragraphs.Format = myDup
```



Paragraphs Collection Object

Multiple objects [└ Paragraphs \(Paragraph\)](#)
└ Multiple objects

A collection of [Paragraph](#) objects in a selection, range, or document.

Using the Paragraphs Collection

Use the **Paragraphs** property to return the **Paragraphs** collection. The following example formats the selected paragraphs to be double-spaced and right-aligned.

```
With Selection.Paragraphs
    .Alignment = wdAlignParagraphRight
    .LineSpacingRule = wdLineSpaceDouble
End With
```

Use the [Add](#), [InsertParagraph](#), [InsertParagraphAfter](#), or [InsertParagraphBefore](#) method to add a new paragraph to a document. The following example adds a new paragraph before the first paragraph in the selection.

```
Selection.Paragraphs.Add Range:=Selection.Paragraphs(1).Range
```

The following example also adds a paragraph before the first paragraph in the selection.

```
Selection.Paragraphs(1).Range.InsertParagraphBefore
```

Use **Paragraphs(index)**, where *index* is the index number, to return a single **Paragraph** object. The following example right aligns the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Alignment = wdAlignParagraphRight
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.

PictureFormat Object

[Shapes \(Shape\)](#) └ [PictureFormat](#)

Contains properties and methods that apply to pictures and OLE objects. The [LinkFormat](#) object contains properties and methods that apply to linked OLE objects only. The [OLEFormat](#) object contains properties and methods that apply to OLE objects whether or not they're linked.

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.

```
With ActiveDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = 0.7
    .ColorType = msoPictureGrayScale
    .CropBottom = 18
End With
```



ProofreadingErrors Collection Object

Multiple objects [↳ ProofreadingErrors \(Range\)](#)
↳ Multiple objects

A collection of spelling and grammatical errors for the specified document or range. There is no ProofreadingError object; instead, each item in the **ProofreadingErrors** collection is a **Range** object that represents one spelling or grammatical error.

Using the ProofreadingErrors Collection

Use the **SpellingErrors** or **GrammaticalErrors** property to return the **ProofreadingErrors** collection. The following example counts the spelling and grammatical errors in the selection and displays the results in a message box.

```
Set pr1 = Selection.Range.SpellingErrors
    sc = pr1.Count
Set pr2 = Selection.Range.GrammaticalErrors
    gc = pr2.Count
Msgbox "Spelling errors: " & sc & vbCr _
    & "Grammatical errors: " & gc
```

Use **SpellingErrors(index)**, where *index* is the index number, to return a single spelling error (represented by a **Range** object). The following example finds the second spelling error in the selection and then selects it.

```
Set myRange = Selection.Range.SpellingErrors(2)
myRange.Select
```

Use **GrammarErrors(index)**, where *index* is the index number, to return a single grammatical error (represented by a **Range** object). The following example returns the sentence that contains the first grammatical error in the selection.

```
Set myRange = Selection.Range.GrammaticalErrors(1)
Msgbox myRange.Text
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object. If all the words in the document or range are spelled correctly and are grammatically correct, the **Count** property for the **ProofreadingErrors** object returns 0 (zero) and the **SpellingChecked** and **GrammarChecked** properties return **True**.

Range Object

Multiple objects  [Range](#)
 Multiple objects

Represents a contiguous area in a document. Each **Range** object is defined by a starting and ending character position. Similar to the way bookmarks are used in a document, **Range** objects are used in Visual Basic procedures to identify specific portions of a document. However, unlike a bookmark, a **Range** object only exists while the procedure that defined it is running.

Note **Range** objects are independent of the selection. That is, you can define and manipulate a range without changing the selection. You can also define multiple ranges in a document, while there can be only one selection per pane.

Using the Range Object

Use the [Range](#) method to return a **Range** object defined by the given starting and ending character positions. The following example returns a **Range** object that refers to the first 10 characters in the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=10)
```

Use the [Range](#) property to return a **Range** object defined by the beginning and end of another object. The **Range** property applies to many objects (for example, **Paragraph**, **Bookmark**, and **Cell**). The following example returns a **Range** object that refers to the first paragraph in the active document.

```
Set aRange = ActiveDocument.Paragraphs(1).Range
```

The following example returns a **Range** object that refers to the second through fourth paragraphs in the active document

```
Set aRange = ActiveDocument.Range( _  
    Start:=ActiveDocument.Paragraphs(2).Range.Start, _  
    End:=ActiveDocument.Paragraphs(4).Range.End)
```

For more information about working with **Range** objects, see [Working with Range Objects](#).



ReadabilityStatistic Object

Multiple objects [↳ ReadabilityStatistics \(ReadabilityStatistic\)](#)

Represents one of the readability statistics for a document or range. The **ReadabilityStatistic** object is a member of the [ReadabilityStatistics](#) collection.

Using the ReadabilityStatistic Object

Use **ReadabilityStatistics**(*index*), where *index* is the index number, to return a single **ReadabilityStatistic** object. The statistics are ordered as follows: Words, Characters, Paragraphs, Sentences, Sentences per Paragraph, Words per Sentence, Characters per Word, Passive Sentences, Flesch Reading Ease, and Flesch-Kincaid Grade Level. The following example returns the character count for the active document.

```
Msgbox ActiveDocument.Content.ReadabilityStatistics(2).Value
```



ReadabilityStatistics Collection Object

Multiple objects [↳ ReadabilityStatistics \(ReadabilityStatistic\)](#)

A collection of [ReadabilityStatistic](#) objects for a document or range.

Using the ReadabilityStatistics Collection

Use the **ReadabilityStatistics** property to return the **ReadabilityStatistics** collection. The following example enumerates the readability statistics for the selection and displays each one in a message box.

```
For each rs in Selection.Range.ReadabilityStatistics
    MsgBox rs.Name & " - " & rs.Value
Next rs
```

Use **ReadabilityStatistics(index)**, where *index* is the index number, to return a single **ReadabilityStatistic** object. The statistics are ordered as follows: Words, Characters, Paragraphs, Sentences, Sentences per Paragraph, Words per Sentence, Characters per Word, Passive Sentences, Flesch Reading Ease, and Flesch-Kincaid Grade Level. The following example returns the word count for the active document.

```
Set myRange = ActiveDocument.Content
wordval = myRange.ReadabilityStatistics(1).Value
MsgBox wordval
```



RecentFile Object

[RecentFiles](#) └ [RecentFile](#)
└ [Document](#)

Represents a recently used file. The **RecentFile** object is a member of the [RecentFiles](#) collection. The **RecentFiles** collection includes all the files that have been used recently. The items in the **RecentFiles** collection are displayed at the bottom of the **File** menu.

Using the RecentFile Object

Use **RecentFiles**(*index*), where *index* is the index number, to return a single **RecentFile** object. The index number represents the position of the file on the **File** menu. The following example opens the first document in the **RecentFiles** collection.

```
If RecentFiles.Count >= 1 Then RecentFiles(1).Open
```

Use the [Add](#) method to add a file to the **RecentFiles** collection. The following example adds the active document to the list of recently-used files.

```
If ActiveDocument.Saved = True Then  
    RecentFiles.Add Document:=ActiveDocument.FullName, _  
        ReadOnly:=True  
End If
```

Remarks

The [SaveAs](#) and [Open](#) methods include an *AddToRecentFiles* argument that controls whether or not a file is added to the recently-used-files list when the file is opened or saved.

RecentFiles Collection Object

Multiple objects [RecentFiles](#)
[RecentFile](#)

A collection of [RecentFile](#) objects that represents the files that have been used recently. The items in the **RecentFiles** collection are displayed at the bottom of the **File** menu.

Using the RecentFiles Collection

Use the [RecentFiles](#) property to return the **RecentFiles** collection. The following example sets five as the maximum number of files that the **RecentFiles** collection can contain.

```
RecentFiles.Maximum = 5
```

Use the [Add](#) method to add a file to the **RecentFiles** collection. The following example adds the active document to the list of recently-used files.

```
If ActiveDocument.Saved = True Then  
    RecentFiles.Add Document:=ActiveDocument.FullName, _  
        ReadOnly:=True  
End If
```

Use **RecentFiles(index)**, where *index* is the index number, to return a single **RecentFile** object. The index number represents the position of the file on the **File** menu. The following example opens the first document in the **RecentFiles** collection.

```
If RecentFiles.Count >= 1 Then RecentFiles(1).Open
```

Remarks

The [SaveAs](#) and [Open](#) methods include an *AddToRecentFiles* argument that controls whether or not a file is added to the recently-used-files list when the file is opened or saved.

Replacement Object

[Find](#)  [Replacement](#)
 Multiple objects

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

Using the Replacement Object

Use the [Replacement](#) property to return a **Replacement** object. The following example replaces the next occurrence of the word "hi" with the word "hello."

```
With Selection.Find
    .Text = "hi"
    .ClearFormatting
    .Replacement.Text = "hello"
    .Replacement.ClearFormatting
    .Execute Replace:=wdReplaceOne, Forward:=True
End With
```

To find and replace formatting, set both the find text and the replace text to empty strings ("") and set the *Format* argument of the [Execute](#) method to **True**. The following example removes all the bold formatting in the active document. The **Bold** property is **True** for the [Find](#) object and **False** for the **Replacement** object.

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Font.Bold = True
    .Text = ""
    With .Replacement
        .ClearFormatting
        .Font.Bold = False
        .Text = ""
    End With
    .Execute Format:=True, Replace:=wdReplaceAll
End With
```



Reviewer Object

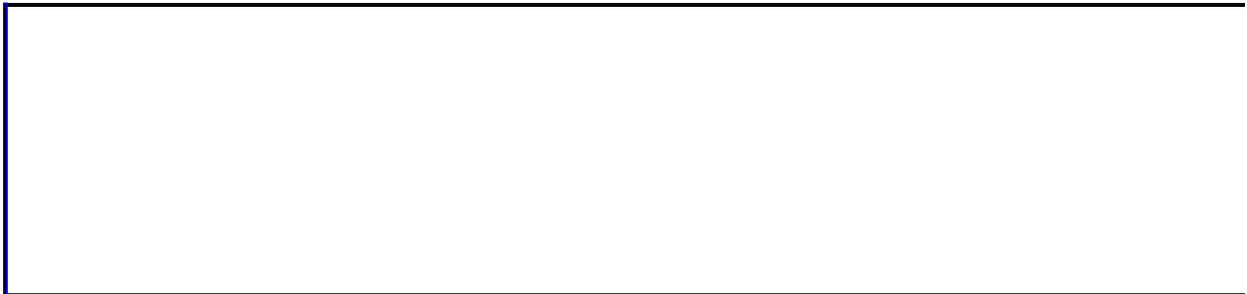
[Reviewers](#) | [Reviewer](#)

Represents a single reviewer of a document in which changes have been tracked. The **Reviewer** object is a member of the [Reviewers](#) collection.

Using the Reviewer object

Use **Reviewers**(*index*), where *index* is the name or number of the reviewer, to return a **Reviewer** object. Use the **Visible** property to display or hide individual reviewers in a document. The following example hides the reviewer named "Jeff Smith" and displays the reviewer named "Judy Lew." This assumes that "Jeff Smith" and "Judy Lew" are members of the **Reviewers** collection. If they are not, you will receive an error.

```
Sub ShowHide()  
  With ActiveWindow.View  
    .Reviewers("Jeff Smith").Visible = False  
    .Reviewers("Judy Lew").Visible = True  
  End With  
End Sub
```



Reviewers Collection

[View](#) [└ Reviewers](#)
[└ Reviewer](#)

A collection of [Reviewer](#) objects that represents the reviewers of one or more documents. The **Reviewers** collection contains the names of all reviewers who have reviewed documents opened or edited on a machine.

Using the Reviewers collection

Use **Reviewers**(*index*), where *index* is the name or index number of the reviewer, to return a single reviewer in the **Reviewers** collection. This example hides revisions made by the first reviewer in the **Reviewers** collection.

```
Sub HideAuthorRevisions(blnRev As Boolean)
    ActiveWindow.View.Reviewers(Index:=1) _
        .Visible = False
End Sub
```



Revision Object

Multiple objects [↳ Revision](#)
↳ Multiple objects

Represents a change marked with a revision mark. The **Revision** object is a member of the [Revisions](#) collection. The **Revisions** collection includes all the revision marks in a range or document.

Using the Revision Object

Use **Revisions**(*index*), where *index* is the index number, to return a single **Revision** object. The index number represents the position of the revision in the range or document. The following example displays the author name for the first revision in section one of the active document.

```
MsgBox ActiveDocument.Sections(1).Range.Revisions(1).Author
```

The **Add** method isn't available for the **Revisions** collection. **Revision** objects are added when change tracking is enabled. Set the [TrackRevisions](#) property to **True** to track revisions made to the document text. The following example enables revision tracking and then inserts "Action " before the selection.

```
ActiveDocument.TrackRevisions = True  
Selection.InsertBefore "Action "
```



Revisions Collection Object

Multiple objects [↳ Revisions \(Revision\)](#)
[↳ Range](#)

A collection of [Revision](#) objects that represent the changes marked with revision marks in a range or document.

Using the Revisions Collection

Use the **Revisions** property to return the **Revisions** collection. The following example displays the number of revisions in the main text story.

```
MsgBox ActiveDocument.Revisions.Count
```

The following example accepts all the revisions in the selection.

```
For Each myRev In Selection.Range.Revisions  
    myRev.Accept  
Next myRev
```

The following example accepts all the revisions in the first paragraph in the selection.

```
Set myRange = Selection.Paragraphs(1).Range  
myRange.Revisions.AcceptAll
```

The **Add** method isn't available for the **Revisions** collection. **Revision** objects are added when change tracking is enabled. Set the [TrackRevisions](#) property to **True** to track revisions made to the document text. The following example enables revision tracking in the active document and then inserts "The " before the selection.

```
ActiveDocument.TrackRevisions = True  
Selection.InsertBefore "The "
```

Use **Revisions(index)**, where *index* is the index number, to return a single **Revision** object. The index number represents the position of the revision in the range or document. The following example displays the author name for the first revision in the first section.

```
MsgBox ActiveDocument.Sections(1).Range.Revisions(1).Author
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.

RoutingSlip Object

[Documents \(Document\)](#)  [RoutingSlip](#)

Represents the routing slip associated with a document. You use a routing slip to send a document through an electronic mail system.

Using the RoutingSlip Object

Use the **RoutingSlip** property to return the **RoutingSlip** object. The following example routes the active document to the specified recipients, one after another.

```
ActiveDocument.HasRoutingSlip = True
With ActiveDocument.RoutingSlip
    .Subject = "Project Documentation"
    .AddRecipient "Don Funk"
    .AddRecipient "Dave Edson"
    .Delivery = wdOneAfterAnother
End With
ActiveDocument.Route
```

Remarks

The **RoutingSlip** object cannot be used (doesn't exist) unless the [HasRoutingSlip](#) property for the document is set to **True**.

Row Object

Multiple objects [↳ Rows \(Row\)](#)
↳ Multiple objects

Represents a row in a table. The **Row** object is a member of the [Rows](#) collection. The **Rows** collection includes all the rows in the specified selection, range, or table.

Using the Row Object

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 selection, range, or table. The following example deletes the first row in the first table in the active document.

```
ActiveDocument.Tables(1).Rows(1).Delete
```

Use the **Add** method to add a row to a table. The following example inserts a row before the first row in the selection.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.Rows.Add BeforeRow:=Selection.Rows(1)  
End If
```

Remarks

Use the **Cells** property to modify the individual cells in a **Row** object. The following example adds a table to the selection and then inserts numbers into each cell in the second row of the table.

```
Selection.Collapse Direction:=wdCollapseEnd
If Selection.Information(wdWithInTable) = False Then
    Set myTable = _
        ActiveDocument.Tables.Add(Range:=Selection.Range, _
            NumRows:=3, NumColumns:=5)
    For Each aCell In myTable.Rows(2).Cells
        i = i + 1
        aCell.Range.Text = i
    Next aCell
End If
```



Rows Collection Object

Multiple objects [↳ Rows \(Row\)](#)
↳ Multiple objects

A collection of [Row](#) objects that represent the table rows in the specified selection, range, or table.

Using the Rows Collection

Use the **Rows** property to return the **Rows** collection. The following example centers rows in the first table in the active document between the left and right margins.

```
ActiveDocument.Tables(1).Rows.Alignment = wdAlignRowCenter
```

Use the [Add](#) method to add a row to a table. The following example inserts a row before the first row in the selection.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.Rows.Add BeforeRow:=Selection.Rows(1)  
End If
```

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 selection, range, or table. The following example deletes the first row in the first table in the active document.

```
ActiveDocument.Tables(1).Rows(1).Delete
```



Section Object

Multiple objects [↳ Sections \(Section\)](#)
↳ Multiple objects

Represents a single section in a selection, range, or document. The **Section** object is a member of the [Sections](#) collection. The **Sections** collection includes all the sections in a selection, range, or document.

Using the Section Object

Use **Sections**(*index*), where *index* is the index number, to return a single **Section** object. The following example changes the left and right page margins for the first section in the active document.

```
With ActiveDocument.Sections(1).PageSetup
    .LeftMargin = InchesToPoints(0.5)
    .RightMargin = InchesToPoints(0.5)
End With
```

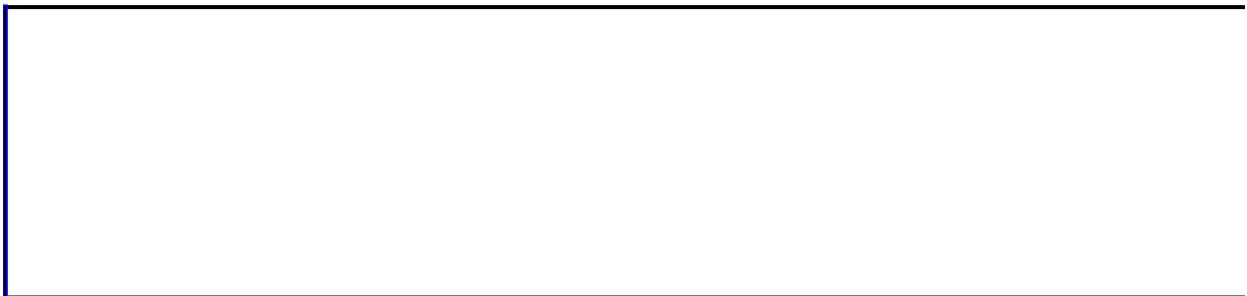
Use the [Add](#) method or the [InsertBreak](#) method to add a new section to a document. The following example adds a new section at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.Sections.Add Range:=myRange
myRange.InsertParagraphAfter
```

The following example adds a section break above the first paragraph in the selection.

```
Selection.Paragraphs(1).Range.InsertBreak _
    Type:=wdSectionBreakContinuous
```

Note The [Headers](#) and [Footers](#) properties of the specified **Section** object return a **HeadersFooters** object.



Sections Collection Object

Multiple objects [↳ Sections \(Section\)](#)
↳ Multiple objects

A collection of [Section](#) objects in a selection, range, or document.

Using the Sections Collection

Use the **Sections** property to return the **Sections** collection. The following example inserts text at the end of the last section in the active document.

```
With ActiveDocument.Sections.Last.Range
    .Collapse Direction:=wdCollapseEnd
    .InsertAfter "end of document"
End With
```

Use the [Add](#) method or the [InsertBreak](#) method to add a new section to a document. The following example adds a new section at the beginning of the active document.

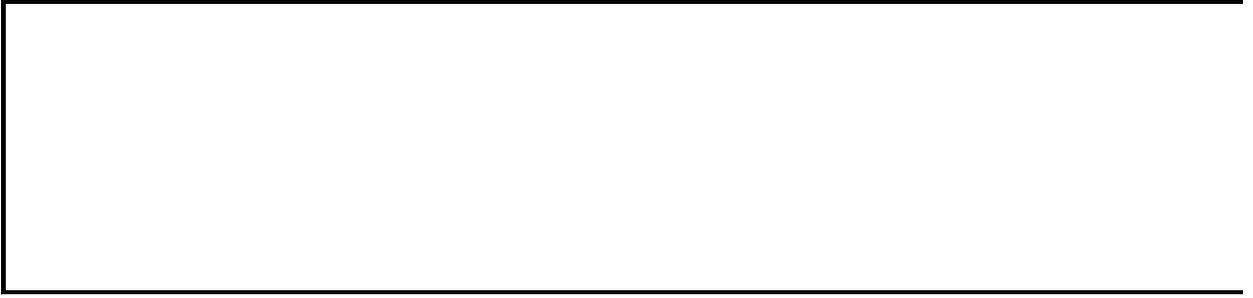
```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.Sections.Add Range:=myRange
myRange.InsertParagraphAfter
```

The following example displays the number of sections in the active document, adds a section break above the first paragraph in the selection, and then displays the number of sections again.

```
MsgBox ActiveDocument.Sections.Count & " sections"
Selection.Paragraphs(1).Range.InsertBreak _
    Type:=wdSectionBreakContinuous
MsgBox ActiveDocument.Sections.Count & " sections"
```

Use **Sections(index)**, where *index* is the index number, to return a single **Section** object. The following example changes the left and right page margins for the first section in the active document.

```
With ActiveDocument.Sections(1).PageSetup
    .LeftMargin = InchesToPoints(0.5)
    .RightMargin = InchesToPoints(0.5)
End With
```



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 document, or it represents the insertion point if nothing in the document is selected. There can only be one **Selection** object per document 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, Word returns the selection from the active pane of the active document window. The following example copies the current selection from the active document.

```
Selection.Copy
```

The following example cuts the selection from the third document in the [Documents](#) collection. The document doesn't have to be active to access its current selection.

```
Documents(3).ActiveWindow.Selection.Cut
```

The following example copies the selection from the first pane of the active document and pastes it into the second pane.

```
ActiveDocument.ActiveWindow.Panes(1).Selection.Copy  
ActiveDocument.ActiveWindow.Panes(2).Selection.Paste
```

The [Text](#) property is the default property of the **Selection** object. Use this property to set or return the text in the current selection. The following example assigns the text in the current selection to the variable `strTemp`, removing the last character if it is a paragraph mark.

```
Dim strTemp as String  
  
strTemp = Selection.Text  
If Right(strTemp, 1) = vbCr Then _  
    strTemp = Left(strTemp, Len(strTemp) - 1)
```

The **Selection** object has various methods and properties with which you can collapse, expand, or otherwise change the current selection. The following example moves the insertion point to the end of the document and selects the last three lines.

```
Selection.EndOf Unit:=wdStory, Extend:=wdMove  
Selection.HomeKey Unit:=wdLine, Extend:=wdExtend  
Selection.MoveUp Unit:=wdLine, Count:=2, Extend:=wdExtend
```

The **Selection** object has various methods and properties with which you can edit selected text in a document. The following example selects the first sentence in the active document and replaces it with a new paragraph.

```
Options.ReplaceSelection = True
ActiveDocument.Sentences(1).Select
Selection.TypeText "Material below is confidential."
Selection.TypeParagraph
```

The following example cuts the last paragraph of the first document in the [Documents](#) collection and pastes it at the beginning of the second document.

```
With Documents(1)
    .Paragraphs.Last.Range.Select
    .ActiveWindow.Selection.Cut
End With

With Documents(2).ActiveWindow.Selection
    .StartOf Unit:=wdStory, Extend:=wdMove
    .Paste
End With
```

The **Selection** object has various methods and properties with which you can change the formatting of the current selection. The following example changes the font of the current selection from Times New Roman to Tahoma.

```
If Selection.Font.Name = "Times New Roman" Then _
    Selection.Font.Name = "Tahoma"
```

Use properties like [Flags](#), [Information](#), and [Type](#) to return information about the current selection. You could use the following example in a procedure to determine if there were anything actually selected in the active document; if not, the rest of the procedure would be skipped.

```
If Selection.Type = wdSelectionIP Then
    MsgBox Prompt:="You haven't selected any text! Exiting procedure"
    Exit Sub
End If
```

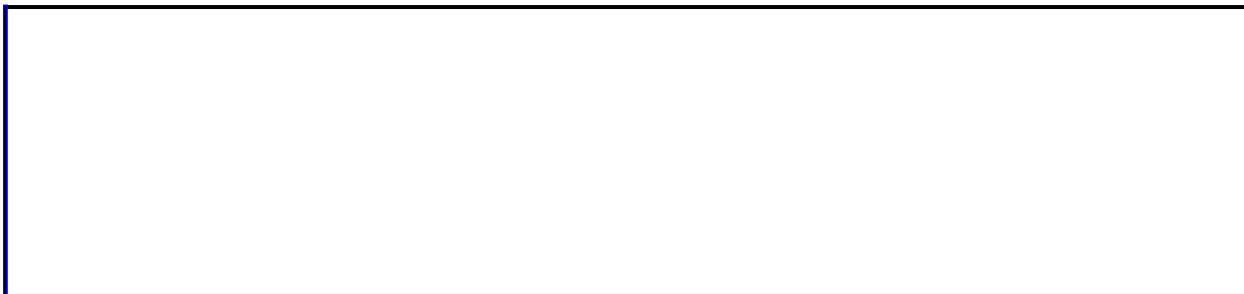
Remarks

Even when a selection is collapsed to an insertion point, it isn't necessarily empty. For example, the [Text](#) property will still return the character to the right of the insertion point; this character also appears in the [Characters](#) collection of the **Selection** object. However, calling methods like [Cut](#) or [Copy](#) from a collapsed selection will cause an error.

It's possible for the user to select a region in a document that doesn't represent contiguous text (for example, when using the ALT key with the mouse). Because the behavior of such a selection can be unpredictable, you may want to include a step in your code that checks the [Type](#) property of a selection before performing any operations on it (`Selection.Type = wdSelectionBlock`). Similarly, selections that include table cells can also lead to unpredictable behavior. The [Information](#) property will tell you if a selection is inside a table (`Selection.Information(wdWithinTable) = True`). The following example determines if a selection is normal (in other words, it isn't a row or column in a table, it isn't a vertical block of text, and so forth); you could use it to test the current selection before performing any operations on it.

```
If Selection.Type <> wdSelectionNormal Then
    MsgBox Prompt:="Not a valid selection! Exiting procedure..."
    Exit Sub
End If
```

Because [Range](#) objects share many of the same methods and properties as **Selection** objects, using **Range** objects is preferable for manipulating a document when there isn't a reason to physically change the current selection. For more information on **Selection** and **Range** objects, see [Working with the Selection object](#) and [Working with Range objects](#).



↳ [Show All](#)

Sentences Collection Object

Multiple objects [└ Sentences \(Range\)](#)
└ Multiple objects

A collection of [Range](#) objects that represent all the sentences in a selection, range, or document. There is no Sentence object.

Using the Sentences Collection

Use the **Sentences** property to return the **Sentences** collection. The following example displays the number of sentences selected.

```
MsgBox Selection.Sentences.Count & " sentences are selected"
```

Use **Sentences(index)**, where *index* is the index number, to return a **Range** object that represents a sentence. The index number represents the position of a sentence in the **Sentences** collection. The following example formats the first sentence in the active document.

```
With ActiveDocument.Sentences(1)  
    .Bold = True  
    .Font.Size = 24  
End With
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main [story](#) only. To count items in other stories use the collection with the **Range** object.

The **Add** method isn't available for the **Sentences** collection. Instead, use the [InsertAfter](#) or [InsertBefore](#) method to add a sentence to a **Range** object. The following example inserts a sentence after the first paragraph in the active document.

```
With ActiveDocument
    MsgBox .Sentences.Count & " sentences"
    .Paragraphs(1).Range.InsertParagraphAfter
    .Paragraphs(2).Range.InsertBefore "The house is blue."
    MsgBox .Sentences.Count & " sentences"
End With
```



Shading Object

Multiple objects [└ Shading](#)

Contains shading attributes for an object.

Using the Shading Object

Use the **Shading** property to return the **Shading** object. The following example applies fine gray shading to the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Shading.Texture = wdTexture10Percent
```

The following example applies shading with different foreground and background colors to the selection.

```
With Selection.Shading
    .Texture = wdTexture20Percent
    .ForegroundColorIndex = wdBlue
    .BackgroundColorIndex = wdYellow
End With
```

The following example applies a vertical line texture to the first row in the first table in the active document.

```
ActiveDocument.Tables(1).Rows(1).Shading.Texture = _
    wdTextureVertical
```



ShadowFormat Object

[Shapes \(Shape\)](#) └ [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 semitransparent, blue shadow is offset 5 points to the right of the rectangle and 3 points above it.

```
With ActiveDocument.Shapes _  
    .AddShape(msoShapeRectangle, 50, 50, 100, 200).Shadow  
    .ForeColor.RGB = RGB(0, 0, 128)  
    .OffsetX = 5  
    .OffsetY = -3  
    .Transparency = 0.5  
    .Visible = True  
End With
```



Shape Object

Multiple objects [└ Shapes \(Shape\)](#)
└ 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 in the main story of a document or in all the headers and footers of a document.

A shape is always attached to an anchoring range. You can position the shape anywhere on the page that contains the anchor.

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, indexed by name or number.
- Return a shape or shapes within a selection.
- Return a newly created shape.
- Return a single shape from within a group.
- Return a newly formed group of shapes.

Returning 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.

```
ActiveDocument.Shapes(1).Flip msoFlipHorizontal
```

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

```
ActiveDocument.Shapes("Rectangle 1").Flip msoFlipHorizontal
```

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.

Returning 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.

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

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

```
Selection.ShapeRange.Fill.ForeColor.RGB = RGB(255, 0, 0)
```

Returning 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](#), [AddCurve](#), [AddLabel](#), [AddLine](#), [AddOleControl](#), [AddOleObject](#), [AddPolyline](#), [AddShape](#), [AddTextbox](#), [AddTextEffect](#), or [BuildFreeForm](#). The following example adds a rectangle to the active document.

```
ActiveDocument.Shapes.AddShape msoShapeRectangle, 50, 50, 100, 200
```

Returning a Single Shape from Within a Group

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.

Returning a Newly Formed Group of Shapes

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.

Anchoring and Positioning a Shape

Every **Shape** object is anchored to a range of text. A shape is anchored to the beginning of the first paragraph that contains the anchoring range. The shape will always remain on the same page as its anchor.

You can view the anchor itself by setting the [ShowObjectAnchors](#) property to **True**. The shape's [Top](#) and [Left](#) properties determine its vertical and horizontal positions. The shape's [RelativeHorizontalPosition](#) and [RelativeVerticalPosition](#) properties determine whether the position is measured from the anchoring paragraph, the column that contains the anchoring paragraph, the margin, or the edge of the page.

If the [LockAnchor](#) property for the shape is set to **True**, you cannot drag the anchor from its position on the page.

Formatting 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 the [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 **WrapFormat** 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.

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.

The **TextFrame** property returns the [TextFrame](#) object, which contains all the properties and methods for attaching text to shapes and linking the text between text frames.

Remarks

Shape objects are anchored to a range of text but are free-floating and can be positioned anywhere on the page. **InlineShape** objects are treated like characters and are positioned as characters within a line of text. You can use the **ConvertToInlineShape** method and the **ConvertToShape** method to convert shapes from one type to the other. You can convert only pictures, OLE objects, and ActiveX controls to inline shapes.



ShapeNode Object

[Shapes \(Shape\)](#) └ [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 three must be a freeform.

```
With ActiveDocument.Shapes(3)
    If .Nodes(1).EditingType = msoEditingCorner Then
        .Nodes.SetEditingType 1, msoEditingSmooth
    End If
End With
```



ShapeNodes Collection Object

[Shapes \(Shape\)](#) └ [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 the **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.

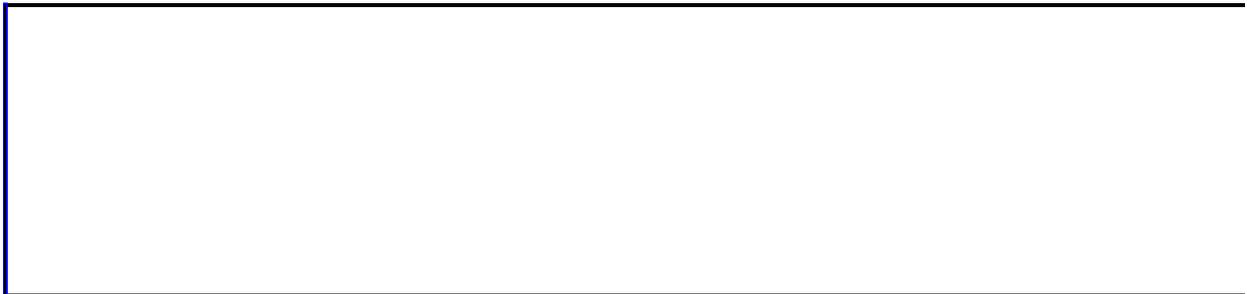
```
ActiveDocument.Shapes(3).Nodes.Delete 4
```

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 be a freeform with at least four nodes.

```
With ActiveDocument.Shapes(3).Nodes  
    .Insert 4, msoSegmentCurve, msoEditingSmooth, 210, 100  
End With
```

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.

```
With ActiveDocument.Shapes(3)  
    If .Nodes(1).EditingType = msoEditingCorner Then  
        .Nodes.SetEditingType 1, msoEditingSmooth  
    End If  
End With
```



ShapeRange Collection Object

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 you specify by name or index number.
- Return a **ShapeRange** object within a selection or range.

Returning a Set of Shapes You Specify by Name or Index Number

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

```
ActiveDocument.Shapes.Range(Array(1, 3)).Fill.Patterned _  
    msoPatternHorizontalBrick
```

The following example selects the shapes named "Oval 4" and "Rectangle 5" on the active document.

```
ActiveDocument.Shapes.Range(Array("Oval 4", "Rectangle 5")).Select
```

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)`.

Returning a ShapeRange Object Within a Selection or Range

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.

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

This example selects all the shapes in the first section of the active document.

```
Set myRange = ActiveDocument.Sections(1).Range  
myRange.ShapeRange.Select
```

Aligning, Distributing, and Grouping 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.

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.

Remarks

The recorder always uses the [ShapeRange](#) property when recording shapes.

A **ShapeRange** object doesn't include [InlineShape](#) objects.



Shapes Collection Object

Multiple objects [└ Shapes \(Shape\)](#)
└ Multiple objects

A collection of [Shape](#) objects that represent all the shapes in a document or all the shapes in all the headers and footers in a document. 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 you want to work with.

Using the Shapes Collection

Use the [Shapes](#) property to return the **Shapes** collection. The following example selects all the shapes on the active document.

```
ActiveDocument.Shapes.SelectAll
```

Note If you want to do something (like delete or set a property) to all the shapes on a document 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](#), [AddCurve](#), [AddLabel](#), [AddLine](#), [AddOleControl](#), [AddOleObject](#), [AddPolyline](#), [AddShape](#), [AddTextbox](#), [AddTextEffect](#), or [BuildFreeForm](#) to add a shape to a document return a **Shape** object that represents the newly created shape The following example adds a rectangle to the active document.

```
ActiveDocument.Shapes.AddShape msoShapeRectangle, 50, 50, 100, 200
```

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.

```
ActiveDocument.Shapes(1).Flip msoFlipHorizontal
```

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

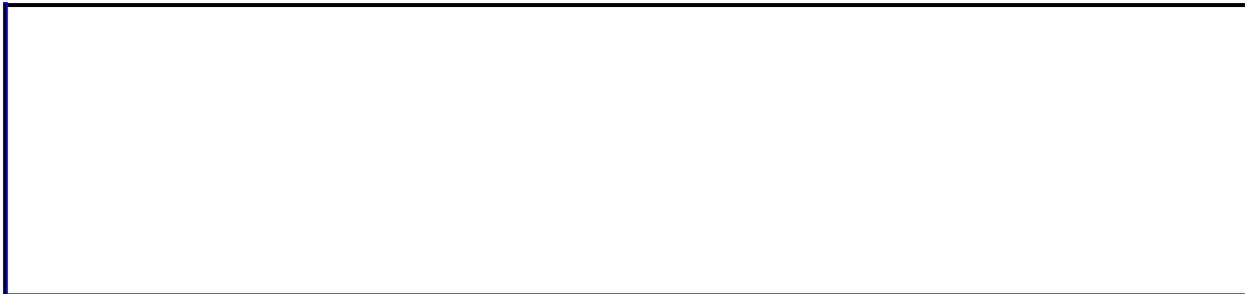
```
ActiveDocument.Shapes("Rectangle 1").Flip msoFlipHorizontal
```

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.

Remarks

The **Shapes** collection does not include [InlineShape](#) objects. **InlineShape** objects are treated like characters and are positioned as characters within a line of text. **Shape** objects are anchored to a range of text but are free-floating and can be positioned anywhere on the page. You can use the [ConvertToInlineShape](#) method and the [ConvertToShape](#) method to convert shapes from one type to the other. You can convert only pictures, OLE objects, and ActiveX controls to inline shapes.

The **Count** property for this collection in a document returns the number of items in the main story only. To count the shapes in all the headers and footers, use the **Shapes** collection with any **HeaderFooter** object.



SmartTag Object

[SmartTags](#) └ [SmartTag](#)
└ Multiple objects

Represents a string in a document or range that contains recognized type information. The **SmartTag** object is a member of the [SmartTags](#) collection. The **SmartTags** collection contains all the smart tags in a document or range of text within a document. Microsoft Word uses a recognizer file to label smart tags, and it uses an action file to execute actions related to the smart tags, such as linking to Web sites.

Using the SmartTag object

Use the [Item](#) method — or [SmartTags\(index\)](#), where *index* represents the number of the smart tag — to return a single **SmartTag** object. This example adds custom properties to the first smart tag in the active document.

```
Sub NewSTProp()  
    ActiveDocument.SmartTags(Index:=1).Properties _  
        .Add Name:="President", Value:=True  
End Sub
```



SmartTags Collection

Multiple objects [SmartTags](#)
[SmartTag](#)

A collection of [SmartTag](#) objects that represents the text in a document that is marked as containing recognized type information. The **SmartTags** collection contains all the smart tags in a document or range of text within a document. Microsoft Word uses a recognizer file to label smart tags, and it uses an action file to execute actions related to the smart tags, such as linking to Web sites.

Using the SmartTags collection

Use the [Item](#) method — or [SmartTags\(index\)](#), where *index* represents the number of the smart tag — to return a single **SmartTag** object. This example adds custom properties to the first smart tag in the active document.

```
Sub NewSmartTagProp()  
    ActiveDocument.SmartTags(1).Properties _  
        .Add Name:="President", Value:=True  
End Sub
```



SpellingSuggestion Object

Multiple objects [↳ SpellingSuggestions \(SpellingSuggestion\)](#)

Represents a single spelling suggestion for a misspelled word. The **SpellingSuggestion** object is a member of the [SpellingSuggestions](#) collection. The **SpellingSuggestions** collection includes all the suggestions for a specified word or for the first word in the specified range.

Using the SpellingSuggestion Object

Use **GetSpellingSuggestions**(*index*), where *index* is the index number, to return a single **SpellingSuggestion** object. The following example checks to see whether there are any spelling suggestions for the first word in the active document. If there are, the first suggestion is displayed in a message box.

```
If ActiveDocument.Words(1).GetSpellingSuggestions.Count <> 0 Then
    MsgBox _
        ActiveDocument.Words(1).GetSpellingSuggestions.Item(1).Name
EndIf
```

Remarks

The **Count** property for the **SpellingSuggestions** object returns 0 (zero) if the word is spelled correctly or if there are no suggestions.

--

SpellingSuggestions Collection Object

Multiple objects [↳ SpellingSuggestions \(SpellingSuggestion\)](#)

A collection of [SpellingSuggestion](#) objects that represent all the suggestions for a specified word or for the first word in the specified range.

Using the SpellingSuggestions Collection

Use the [GetSpellingSuggestions](#) method to return the **SpellingSuggestions** collection. The **SpellingSuggestions** method, when applied to the **Application** object, must specify the word to be checked. When the **GetSpellingSuggestions** method is applied to a range, the first word in the range is checked. The following example checks to see whether there are any spelling suggestions for any of the words in the active document. If there are, the suggestions are displayed in message boxes.

```
For Each wd In ActiveDocument.Words
    Set sugg = wd.GetSpellingSuggestions
    If sugg.Count <> 0 Then
        For Each ss In sugg
            MsgBox ss.Name
        Next ss
    End If
Next wd
```

Remarks

You cannot add suggestions to or remove suggestions from the collection of spelling suggestions. Spelling suggestions are derived from main and custom dictionary files.

↳ [Show All](#)

StoryRanges Collection Object

[Documents \(Document\)](#) [StoryRanges \(Range\)](#)

└ Multiple objects

A collection of [Range](#) objects that represent [stories](#) in a document.

Using the StoryRanges Collection

Use the **StoryRanges** property to return the **StoryRanges** collection. The following example removes manual character formatting from the text in all stories other than the main text story in the active document.

```
For Each aStory In ActiveDocument.StoryRanges
    If aStory.StoryType <> wdMainTextStory Then aStory.Font.Reset
Next aStory
```

The **Add** method isn't available for the **StoryRanges** collection. The number of stories in the **StoryRanges** collection is finite.

Use **StoryRanges(index)**, where *index* is a **WdStoryType** constant, to return a single story as a **Range** object. The following example adds text to the primary header story and then displays the text.

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range _
    .Text = "Header text"
MsgBox ActiveDocument.StoryRanges(wdPrimaryHeaderStory).Text
```

The following example copies the text of the footnotes from the active document into a new document.

```
If ActiveDocument.Footnotes.Count >= 1 Then
    ActiveDocument.StoryRanges(wdFootnotesStory).Copy
    Documents.Add.Content.Paste
End If
```

Remarks

If you attempt to return a story that isn't available in the specified document, an error occurs. The following example determines whether or not a footnote story is available in the active document.

```
On Error GoTo errhandler
Set MyRange = ActiveDocument.StoryRanges(wdFootnotesStory)
errhandler:
If Err = 5941 Then MsgBox "The footnotes story is not available."
```

Use the **NextStoryRange** property to loop through all stories in a document. The following example searches each story in the active document for the text "Microsoft Word." When the text is found, it's formatted as italic.

```
For Each myStoryRange In ActiveDocument.StoryRanges
    myStoryRange.Find.Execute _
        FindText:="Microsoft Word", Forward:=True
    While myStoryRange.Find.Found
        myStoryRange.Italic = True
        myStoryRange.Find.Execute _
            FindText:="Microsoft Word", Forward:=True
    Wend
    While Not (myStoryRange.NextStoryRange Is Nothing)
        Set myStoryRange = myStoryRange.NextStoryRange
        myStoryRange.Find.Execute _
            FindText:="Microsoft Word", Forward:=True
        While myStoryRange.Find.Found
            myStoryRange.Italic = True
            myStoryRange.Find.Execute _
                FindText:="Microsoft Word", Forward:=True
        Wend
    Wend
Next myStoryRange
```



Style Object

Multiple objects [└ Styles \(Style\)](#)
└ Multiple objects

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

Using the Style Object

Use `Styles(index)`, where *index* is the style name, a `WdBuiltinStyle` constant or index number, to return a single **Style** object. You must exactly match the spelling and spacing of the style name, but not necessarily its capitalization. The following example modifies the font name of the user-defined style named "Color" in the active document.

```
ActiveDocument.Styles("Color").Font.Name = "Arial"
```

The following example sets the built-in Heading 1 style to not be bold.

```
ActiveDocument.Styles(wdStyleHeading1).Font.Bold = False
```

The style index number represents the position of the style in the alphabetically sorted list of style names. Note that `Styles(1)` is the first style in the alphabetic list. The following example displays the base style and style name of the first style in the **Styles** collection.

```
MsgBox "Base style= " & _  
    & ActiveDocument.Styles(1).BaseStyle & vbCrLf & _  
    & "Style name= " & ActiveDocument.Styles(1).NameLocal
```

To apply a style to a range, paragraph, or multiple paragraphs, set the **Style** property to a user-defined or built-in style name. The following example applies the Normal style to the first four paragraphs in the active document.

```
Set myRange = ActiveDocument.Range( _  
    Start:=ActiveDocument.Paragraphs(1).Range.Start, _  
    End:=ActiveDocument.Paragraphs(4).Range.End)  
myRange.Style = wdStyleNormal
```

The following example applies the Heading 1 style to the first paragraph in the selection.

```
Selection.Paragraphs(1).Style = wdStyleHeading1
```

The following example creates a character style named "Bolded" and applies it to the selection.

```
Set myStyle = ActiveDocument.Styles.Add(Name:="Bolded", _  
    Type:=wdStyleTypeCharacter)  
myStyle.Font.Bold = True  
Selection.Range.Style = "Bolded"
```

Remarks

Use the [OrganizerCopy](#) method to copy styles between documents and templates. Use the [UpdateStyles](#) method to update the styles in the active document to match the style definitions in the attached template. Use the [OpenAsDocument](#) method to open a template as a document so that you can modify the template styles.

Styles Collection Object

[Documents \(Document\)](#) [└ Styles \(Style\)](#)

└ Multiple objects

A collection of [Style](#) objects that represent both the built-in and user-defined styles in a document.

Using the Styles Collection

Use the **Styles** property to return the **Styles** collection. The following example deletes all user-defined styles in the active document.

```
For Each sty In ActiveDocument.Styles
    If sty.BuiltIn = False Then sty.Delete
Next sty
```

Use the **Add** method to create a new user-defined style and add it to the **Styles** collection. The following example adds a new character style named "Introduction" and makes it 12-point Arial, with bold and italic formatting. The example then applies this new character style to the selection.

```
Set myStyle = ActiveDocument.Styles.Add(Name:="Introduction", _
    Type:=wdStyleTypeCharacter)
With myStyle.Font
    .Bold = True
    .Italic = True
    .Name = "Arial"
    .Size = 12
End With
Selection.Range.Style = "Introduction"
```

Use **Styles(index)**, where *index* is the style name, a **WdBuiltinStyle** constant or index number, to return a single **Style** object. You must exactly match the spelling and spacing of the style name, but not necessarily its capitalization. The following example modifies the font of the user-defined style named "Color" in the active document.

```
ActiveDocument.Styles("Color").Font.Name = "Arial"
```

The following example sets the built-in Heading 1 style to not be bold.

```
ActiveDocument.Styles(wdStyleHeading1).Font.Bold = False
```

The style index number represents the position of the style in the alphabetically sorted list of style names. Note that **Styles(1)** is the first style in the alphabetic list. The following example displays the base style and style name of the first style in the **Styles** collection.

```
MsgBox "Base style= " _  
    & ActiveDocument.Styles(1).BaseStyle & vbCrLf _  
    & "Style name= " & ActiveDocument.Styles(1).NameLocal
```

Remarks

The **Styles** object isn't available from the **Template** object. However, you can use the [OpenAsDocument](#) method to open a template as a document so that you can modify styles in the template. The following example changes the formatting of the Heading 1 style in the template attached to the active document.

```
Set aDoc = ActiveDocument.AttachedTemplate.OpenAsDocument
With aDoc
    .Styles(wdStyleHeading1).Font.Name = "Arial"
    .Close SaveChanges:=wdSaveChanges
End With
```

Use the [OrganizerCopy](#) method to copy styles between documents and templates. Use the [UpdateStyles](#) method to update the styles in the active document to match the style definitions in the attached template.



StyleSheet Object

[StyleSheets](#) | [StyleSheet](#)

Represents a single cascading style sheet attached to a web document. The **StyleSheet** object is a member of the [StyleSheets](#) collection. The **StyleSheets** collection contains all the cascading style sheets attached to a specified document.

Using the StyleSheet object

Use the [Item](#) method or `StyleSheets(index)`, where *index* is the name or number of the style sheet, of the `StyleSheets` collection to return a `StyleSheet` object. The following example removes the second style sheet from the `StyleSheets` collection.

```
Sub WebStyleSheets()  
    ActiveDocument.StyleSheets.Item(2).Delete  
End Sub
```

Use the [Index](#) property to determine the precedence of cascading style sheets. The following example creates a table of attached cascading style sheets, ordered and indexed according to which style sheet is most important.

```
Sub CSSTable()  
    Dim styCSS As StyleSheet  
  
    With ActiveDocument.Range(Start:=0, End:=0)  
        .InsertAfter "CSS Name" & vbTab & "Index"  
        .InsertParagraphAfter  
        For Each styCSS In ActiveDocument.StyleSheets  
            .InsertAfter styCSS.Name & vbTab & styCSS.Index  
            .InsertParagraphAfter  
        Next styCSS  
        .ConvertToTable  
    End With  
End Sub
```

Use the [Move](#) method to reorder the precedence of attached style sheets. The following example moves the most important style sheet to the least important of all attached cascading style sheets.

```
Sub MoveCSS()  
    ActiveDocument.StyleSheets(1) _  
        .Move wdStyleSheetPrecedenceLowest  
End Sub
```



StyleSheets Collection

[Document](#) └ [StyleSheets](#)
└ [StyleSheet](#)

A collection of [StyleSheet](#) objects that represents the cascading style sheets attached to a document. The **StyleSheets** collection includes all cascading style sheets displayed in the **Linked CSS Style Sheets** dialog box, accessed using the **Templates and Add-ins** command (**Tools** menu).

Using the StyleSheets collection

Use the [StyleSheets](#) property to return the **StyleSheets** collection. Use the [Add](#) method to add a style sheet to the **StyleSheets** collection. The following example adds three cascading style sheets to the active document and sets the third as the highest in precedence.

```
Sub AddCSS()  
  With ActiveDocument.StyleSheets  
    .Add FileName:="Web.css", Title:="Web Styles"  
    .Add FileName:="New.css", Linktype:=wdStyleSheetLinkTypeImpo  
      Title:="New Styles"  
    .Add FileName:="Defs.css", Title:="Definitions", _  
      Precedence:=wdStyleSheetPrecedenceHighest  
  End With  
End Sub
```



Subdocument Object

[Subdocuments](#)  [Subdocument](#)
 Multiple objects

Represents a subdocument within a document or range. The **Subdocument** object is a member of the [Subdocuments](#) collection. The **Subdocuments** collection includes all the subdocuments in the a range or document.

Using the Subdocument Object

Use **Subdocuments**(*index*), where *index* is the index number, to return a single **Subdocument** object. The following example displays the path and file name of the first subdocument in the active document.

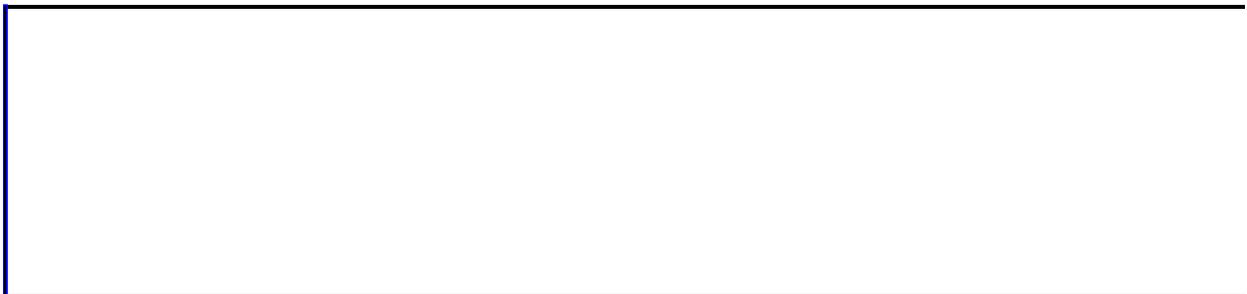
```
If ActiveDocument.Subdocuments(1).HasFile = True Then
    MsgBox ActiveDocument.Subdocuments(1).Path & _
        Application.PathSeparator & _
        ActiveDocument.Subdocuments(1).Name
End If
```

Use the [AddFromFile](#) or [AddFromRange](#) method to add a subdocument to a document. The following example adds a subdocument named "Setup.doc" at the end of the active document.

```
ActiveDocument.Subdocuments.Expanded = True
Selection.EndKey Unit:=wdStory
Selection.InsertParagraphBefore
ActiveDocument.Subdocuments.AddFromFile Name:="C:\Temp\Setup.doc"
```

The following example applies the Heading 1 style to the first paragraph in the selection and then creates a subdocument for the contents of the selection.

```
Selection.Paragraphs(1).Style = wdStyleHeading1
With ActiveDocument.Subdocuments
    .Expanded = True
    .AddFromRange Range:=Selection.Range
End With
```



Subdocuments Collection Object

Multiple objects [↳ Subdocuments \(Subdocument\)](#)
[↳ Range](#)

A collection of [Subdocument](#) objects that represent the subdocuments in a range or document.

Using the Subdocuments Collection

Use the **Subdocuments** property to return the **Subdocuments** collection. The following example expands all the subdocuments in the active document.

```
ActiveDocument.Subdocuments.Expanded = True
```

Use the [AddFromFile](#) or [AddFromRange](#) method to add a subdocument to a document. The following example adds a subdocument named "Setup.doc" at the end of the active document.

```
ActiveDocument.Subdocuments.Expanded = True
Selection.EndKey Unit:=wdStory
Selection.InsertParagraphBefore
ActiveDocument.Subdocuments.AddFromFile Name:="C:\Temp\Setup.doc"
```

The following example applies the Heading 1 style to the first paragraph in the selection and then creates a subdocument for the contents of the selection.

```
Selection.Paragraphs(1).Style = wdStyleHeading1
With ActiveDocument.Subdocuments
    .Expanded = True
    .AddFromRange Range:=Selection.Range
End With
```

Use **Subdocuments(index)**, where *index* is the index number, to return a single **Subdocument** object. The following example displays the path and file name of the first subdocument in the active document.

```
If ActiveDocument.Subdocuments(1).HasFile = True Then
    MsgBox ActiveDocument.Subdocuments(1).Path & _
        Application.PathSeparator _
        & ActiveDocument.Subdocuments(1).Name
End If
```



SynonymInfo Object

Multiple objects [↳ SynonymInfo](#)

Represents the information about synonyms, antonyms, related words, or related expressions for the specified range or a given string.

Using the **SynonymInfo** Object

Use the **SynonymInfo** property to return a **SynonymInfo** object. The **SynonymInfo** object can be returned either from a range or from **Word**. If it's returned from **Word**, you specify the lookup word or phrase and a proofing language ID. If it's returned from a range, **Word** uses the specified range as the lookup word. The following example returns a **SynonymInfo** object from **Word**.

```
temp = SynonymInfo(Word:="meant", LanguageID:=wdEnglishUS).Found
```

The following example returns a **SynonymInfo** object from a range.

```
temp = Selection.Range.SynonymInfo.Found
```

The **Found** property, used in the preceding examples, returns **True** if any information is found in the thesaurus for the specified range or for **Word**. Note, however, that this property returns **True** not only if synonyms are found but also if related words, related expressions, or antonyms are found.

Many of the properties of the **SynonymInfo** object return a **Variant** that contains an array of strings. When working with these properties, you can assign the returned array to a variable and then index the variable to see the elements in the array. In the following example, **Slist** is assigned the synonym list for the first meaning of the selected word or phrase. The **UBound** function finds the upper bound of the array, and then each element is displayed in a message box.

```
Slist = Selection.Range.SynonymInfo.SynonymList(1)
For i = 1 To UBound(Slist)
    MsgBox Slist(i)
Next i
```

You can check the value of the **MeaningCount** property to prevent potential errors in your code. The following example returns a list of synonyms for the second meaning for the word or phrase in the selection and displays these synonyms in the **Immediate** pane.

```
Set synInfo = Selection.Range.SynonymInfo
If synInfo.MeaningCount >= 2 Then
    synList = synInfo.SynonymList(2)
    For i = 1 To UBound(synList)
```

```
        Debug.Print synList(i)
    Next i
Else
    MsgBox "There is no second meaning for the selection."
End If
```



System Object

[Application](#)  [System](#)

Contains information about the computer system.

Using the System Object

Use the **System** property to return the **System** object. If the operating system is Windows, the following example makes a network connection to \\Project\Info.

```
If System.OperatingSystem = "Windows" Then
    System.Connect Path:="\\Project\Info"
End If
```

The following example displays the current screen resolution (for example, "1024 x 768").

```
horz = System.HorizontalResolution
vert = System.VerticalResolution
MsgBox "Resolution = " & horz & " x " & vert
```



Table Object

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

Represents a single table. The **Table** object is a member of the [Tables](#) collection. The **Tables** collection includes all the tables in the specified selection, range, or document.

Using the Table Object

Use **Tables**(*index*), where *index* is the index number, to return a single **Table** object. The index number represents the position of the table in the selection, range, or document. The following example converts the first table in the active document to text.

```
ActiveDocument.Tables(1).ConvertToText Separator:=wdSeparateByTabs
```

Use the **Add** method to add a table at the specified range. The following example adds a 3x4 table at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)  
ActiveDocument.Tables.Add Range:=myRange, NumRows:=3, NumColumns:=4
```


TableOfAuthorities Object

[Documents \(Document\)](#) [└ TablesOfAuthorities \(TableOfAuthorities\)](#)
[└ Range](#)

Represents a single table of authorities in a document (a TOA field). The **TableOfAuthorities** object is a member of the [TablesOfAuthorities](#) collection. The **TablesOfAuthorities** collection includes all the tables of authorities in a document.

Using the TableOfAuthorities Object

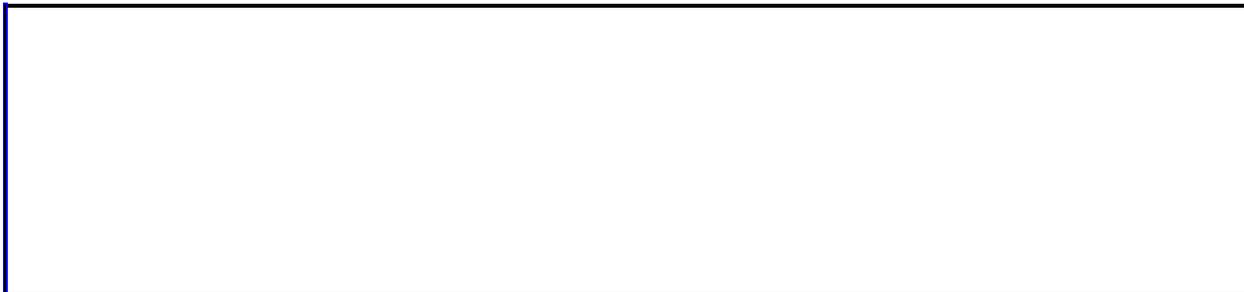
Use **TablesOfAuthorities**(*index*), where *index* is the index number, to return a single **TableOfAuthorities** object. The index number represents the position of the table of authorities in the document. The following example includes category headers in the first table of authorities in the active document and then updates the table.

```
With ActiveDocument.TablesOfAuthorities(1)
    .IncludeCategoryHeader = True
    .Update
End With
```

Use the [Add](#) method to add a table of authorities to a document. The following example adds a table of authorities that includes all categories at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.TablesOfAuthorities.Add Range:=myRange, _
    Passim:=True, Category:=0, EntrySeparator:=", "
```

Note A table of authorities is built from TA (Table of Authorities Entry) fields in a document. Use the [MarkCitation](#) method to mark citations to be included in a table of authorities.



TableOfAuthoritiesCategory Object

[Documents \(Document\)](#) | [TablesOfAuthoritiesCategories \(TablesOfAuthoritiesCategory\)](#)

Represents a single table of authorities category. The **TableOfAuthoritiesCategories** object is a member of the [TablesOfAuthoritiesCategories](#) collection. The **TablesOfAuthoritiesCategories** collection includes all 16 categories listed in the **Category** box on the **Table of Authorities** tab in the **Index and Tables** dialog box (**Insert** menu).

Using the TableOfAuthoritiesCategory Object

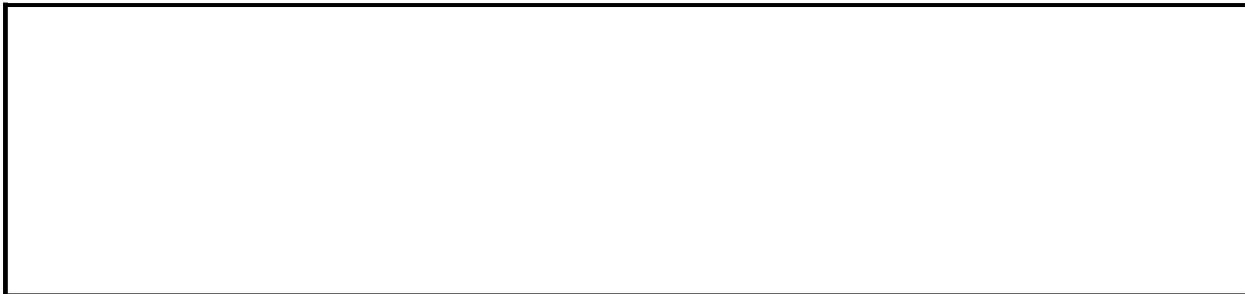
Use **TablesOfAuthoritiesCategories**(*index*), where *index* is the category name or index number, to return a single **TableOfAuthoritiesCategory** object. The following example renames the Rules category as Other Provisions.

```
ActiveDocument.TablesOfAuthoritiesCategories("Rules").Name = _  
    "Other Provisions"
```

The index number represents the position of the category in the **Index and Tables** dialog box (**Insert** menu). The following example displays the name of the first category in the **TablesOfAuthoritiesCategories** collection.

```
MsgBox ActiveDocument.TablesOfAuthoritiesCategories(1).Name
```

The **Add** method isn't available for the **TablesOfAuthoritiesCategories** collection. The collection is limited to 16 items; however, you can use the **Name** property to rename an existing category.



TableOfContents Object

[Documents \(Document\)](#) [└ TablesOfContents \(TableOfContents\)](#)
└ Multiple objects

Represents a single table of contents in a document. The **TableOfContents** object is a member of the [TablesOfContents](#) collection. The **TablesOfContents** collection includes all the tables of contents in a document.

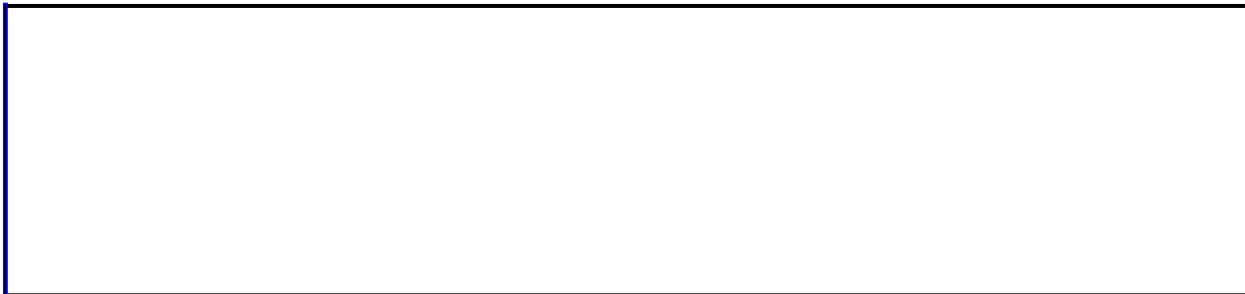
Using the TableOfContents Object

Use **TablesOfContents**(*index*), where *index* is the index number, to return a single **TableOfContents** object. The index number represents the position of the table of contents in the document. The following example updates the page numbers of the items in the first table of figures in the active document.

```
ActiveDocument.TablesOfContents(1).UpdatePageNumbers
```

Use the [Add](#) method to add a table of contents to a document. The following example adds a table of contents at the beginning of the active document. The example builds the table of contents from all paragraphs styled as either Heading 1, Heading 2, or Heading 3.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.TablesOfContents.Add Range:=myRange, _
    UseFields:=False, UseHeadingStyles:=True, _
    LowerHeadingLevel:=3, _
    UpperHeadingLevel:=1
```



TableOfFigures Object

[Documents \(Document\)](#) [└ TablesOfFigures \(TableOfFigures\)](#)
└ Multiple objects

Represents a single table of figures in a document. The **TableOfFigures** object is a member of the [TablesOfFigures](#) collection. The **TablesOfFigures** collection includes all the tables of figures in a document.

Using the TableOfFigures Object

Use **TablesOfFigures**(*index*), where *index* is the index number, to return a single **TableOfFigures** object. The index number represents the position of the table of figures in the document. The following example updates the page numbers of the items in the first table of figures in the active document.

```
ActiveDocument.TablesOfFigures(1).UpdatePageNumbers
```

Use the **Add** method to add a table of figures to a document. A table of figures lists figure captions in the order in which they appear in the document. The following example replaces the selection in the active document with a table of figures that includes caption labels and page numbers.

```
ActiveDocument.TablesOfFigures.Add Range:=Selection.Range, _  
    IncludeLabel:=True, IncludePageNumbers:=True
```



Tables Collection Object

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

A collection of [Table](#) objects that represent the tables in a selection, range, or document.

Using the Tables Collection

Use the **Tables** property to return the **Tables** collection. The following example applies a border around each of the tables in the active document.

```
For Each aTable In ActiveDocument.Tables
    aTable.Borders.OutsideLineStyle = wdLineStyleSingle
    aTable.Borders.OutsideLineWidth = wdLineWidth025pt
    aTable.Borders.InsideLineStyle = wdLineStyleNone
Next aTable
```

Use the [Add](#) method to add a table at the specified range. The following example adds a 3x4 table at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.Tables.Add Range:=myRange, NumRows:=3, NumColumns:=4
```

Use **Tables(index)**, where *index* is the index number, to return a single **Table** object. The index number represents the position of the table in the selection, range, or document. The following example converts the first table in the active document to text.

```
ActiveDocument.Tables(1).ConvertToText Separator:=wdSeparateByTabs
```

Remarks

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object.

TablesOfAuthorities Collection Object

[Document](#)  [TablesOfAuthorities](#)

 Multiple objects

A collection of [TableOfAuthorities](#) objects (TOA fields) that represents the tables of authorities in a document.

Using the TablesOfAuthorities Collection

Use the [TablesOfAuthorities](#) property to return the **TablesOfAuthorities** collection. The following example applies the Classic built-in format to all the tables of authorities in the active document.

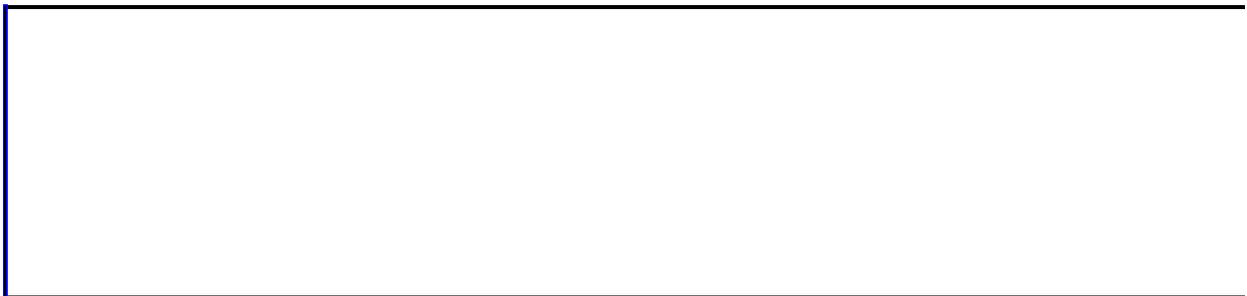
```
ActiveDocument.TablesOfAuthorities.Format = wdTOAClassic
```

Use the [Add](#) method to add a table of authorities to a document. A table of authorities is built from TA (Table of Authorities Entry) fields in a document. The following example adds a table of authorities that includes all categories at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.TablesOfAuthorities.Add Range:=myRange, _
    Passim:=True, Category:=0, EntrySeparator:= ", "
```

Use **TablesOfAuthorities(index)**, where *index* is the index number, to return a single **TableOfAuthorities** object. The index number represents the position of the table of authorities in the document. The following example includes category headers in the first table of authorities in the active document and then updates the table.

```
With ActiveDocument.TablesOfAuthorities(1)
    .IncludeCategoryHeader = True
    .Update
End With
```



TablesOfAuthoritiesCategories Collection Object

[Documents \(Document\)](#) | [TablesOfAuthoritiesCategories \(TablesOfAuthoritiesCategory\)](#)

A collection of [TableOfAuthoritiesCategory](#) objects that represent the table of authorities categories, such as Cases and Statutes. The **TablesOfAuthoritiesCategories** collection includes all 16 categories listed in the **Category** box on the **Table of Authorities** tab in the **Index and Tables** dialog box (**Insert** menu).

Using the TablesOfAuthoritiesCategories Collection

Use the **TablesOfAuthoritiesCategories** property to return the **TablesOfAuthoritiesCategories** collection. The following example displays the names of the categories in the **TablesOfAuthoritiesCategories** collection.

```
For Each aCat In ActiveDocument.TablesOfAuthoritiesCategories
    response = MsgBox(Prompt:=aCat, Buttons:=vbOKCancel)
    If response = vbCancel Then Exit For
Next aCat
```

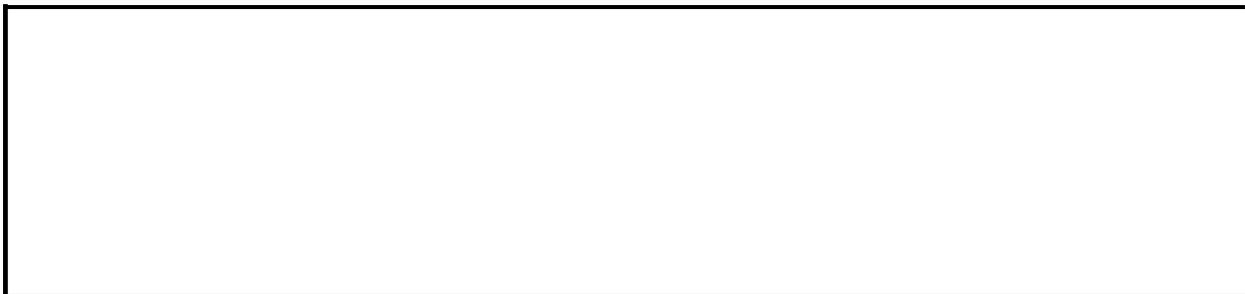
The **Add** method isn't available for the **TablesOfAuthoritiesCategories** collection. The collection is limited to 16 items; however, you can use the **Name** property to rename an existing category.

Use **TablesOfAuthoritiesCategories(index)**, where *index* is the category name or index number, to return a single **TableOfAuthoritiesCategory** object. The following example renames the Rules category as Other Provisions.

```
ActiveDocument.TablesOfAuthoritiesCategories("Rules").Name = _
    "Other Provisions"
```

The index number represents the position of the category in the **Index and Tables** dialog box (**Insert** menu). The following example displays the name of the first category in the **TablesOfAuthoritiesCategories** collection.

```
MsgBox ActiveDocument.TablesOfAuthoritiesCategories(1).Name
```



TablesOfContents Collection Object

[Documents \(Document\)](#) [└ TablesOfContents \(TableOfContents\)](#)
└ Multiple objects

A collection of [TableOfContents](#) objects that represent the tables of contents in a document.

Using the TablesOfContents Collection

Use the **TablesOfContents** property to return the **TablesOfContents** collection. The following example inserts a table of contents entry that references the selected text in the active document.

```
ActiveDocument.TablesOfContents.MarkEntry Range:=Selection.Range, _  
    Level:=2, Entry:="Introduction"
```

Use the [Add](#) method to add a table of contents to a document. The following example adds a table of contents at the beginning of the active document. The example builds the table of contents from all paragraphs styled as either Heading 1, Heading 2, or Heading 3.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)  
ActiveDocument.TablesOfContents.Add Range:=myRange, _  
    UseFields:=False, UseHeadingStyles:=True, _  
    LowerHeadingLevel:=3, _  
    UpperHeadingLevel:=1
```

Use **TablesOfContents(index)**, where *index* is the index number, to return a single **TableOfContents** object. The index number represents the position of the table of contents in the document. The following example updates the page numbers of the items in the first table of figures in the active document.

```
ActiveDocument.TablesOfContents(1).UpdatePageNumbers
```



TablesOfFigures Collection Object

[Documents \(Document\)](#) [└ TablesOfFigures \(TableOfFigures\)](#)

└ Multiple objects

A collection of [TableOfFigures](#) objects that represent the tables of figures in a document.

Using the TablesOfFigures Collection

Use the **TablesOfFigures** property to return the **TablesOfFigures** collection. The following example applies the Classic format to all tables of figures in the active document.

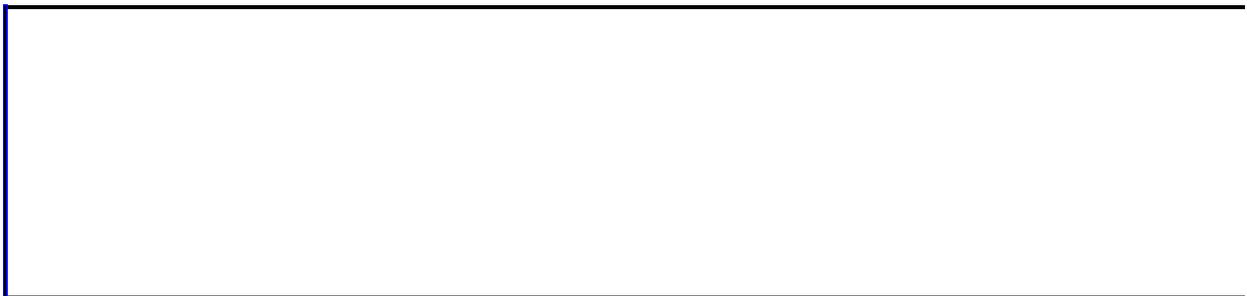
```
ActiveDocument.TablesOfFigures.Format = wdTOFClassic
```

Use the [Add](#) method to add a table of figures to a document. A table of figures lists figure captions in the order in which they appear in the document. The following example replaces the selection in the active document with a table of figures that includes caption labels and page numbers.

```
ActiveDocument.TablesOfFigures.Add Range:=Selection.Range, _  
    IncludeLabel:=True, IncludePageNumbers:=True
```

Use **TablesOfFigures(index)**, where *index* is the index number, to return a single **TableOfFigures** object. The index number represents the position of the table of figures in the document. The following example updates the page numbers of the items in the first table of figures in the active document.

```
ActiveDocument.TablesOfFigures(1).UpdatePageNumbers
```



TableStyle Object

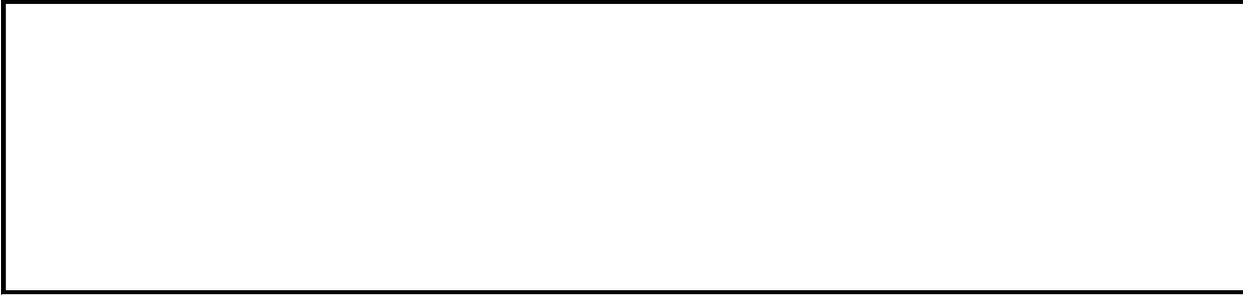
[Style](#) └ [TableStyle](#)
└ Multiple objects

Represents a single style that can be applied to a table.

Using the TableStyle object

Use the [Table](#) property of the [Styles](#) object to return a **TableStyle** object. Use the [Borders](#) property to apply borders to an entire table. Use the [Condition](#) method to apply borders or shading only to specified sections of a table. This example creates a new table style and formats the table with a surrounding border. Special borders and shading are applied to the first and last rows and the last column.

```
Sub NewTableStyle()  
    Dim styTable As Style  
  
    Set styTable = ActiveDocument.Styles.Add( _  
        Name:="TableStyle 1", Type:=wdStyleTypeTable)  
  
    With styTable.Table  
  
        'Apply borders around table  
        .Borders(wdBorderTop).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderBottom).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderLeft).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderRight).LineStyle = wdLineStyleSingle  
  
        'Apply a double border to the heading row  
        .Condition(wdFirstRow).Borders(wdBorderBottom) _  
            .LineStyle = wdLineStyleDouble  
  
        'Apply a double border to the last column  
        .Condition(wdLastColumn).Borders(wdBorderLeft) _  
            .LineStyle = wdLineStyleDouble  
  
        'Apply shading to last row  
        .Condition(wdLastRow).Shading _  
            .BackgroundColor = wdColorGray125  
  
    End With  
  
End Sub
```



TabStop Object

Multiple objects [↳ 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 **TabStops**(*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.

```
Selection.Paragraphs.TabStops(1).Clear
```

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

```
Selection.Paragraphs.TabStops(InchesToPoints(2)) _  
    .Alignment = wdAlignTabRight
```

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.

```
With Selection.Paragraphs.TabStops  
    .Add Position:=InchesToPoints(1), _  
        Leader:=wdTabLeaderDots, Alignment:=wdAlignTabLeft  
    .Add Position:=InchesToPoints(2), Alignment:=wdAlignTabCenter  
End With
```

You can also add a tab stop by specifying a location with the **TabStops** property. The following example adds a right-aligned tab stop positioned at 2 inches to the selected paragraphs.

```
Selection.Paragraphs.TabStops(InchesToPoints(2)) _  
    .Alignment = wdAlignTabRight
```

Note Set the [DefaultTabStop](#) property to adjust the spacing of default tab stops.



TabStops Collection Object

Multiple objects [└ 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 **TabStops** property to return the **TabStops** collection. The following example clears all the custom tab stops from the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).TabStops.ClearAll
```

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.

```
Selection.Paragraphs.TabStops.Add Position:=InchesToPoints(2.5)
For Each aTab In Selection.Paragraphs.TabStops
    MsgBox "Position = " & _
        & PointsToInches(aTab.Position) & " inches"
Next aTab
```

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.

```
With Selection.Paragraphs.TabStops
    .Add Position:=InchesToPoints(1), _
        Leader:=wdTabLeaderDots, Alignment:=wdAlignTabLeft
    .Add Position:=InchesToPoints(2), Alignment:=wdAlignTabCenter
End With
```

You can also add a tab stop by specifying a location with the **TabStops** property. The following example adds a right-aligned tab stop positioned at 2 inches to the selected paragraphs.

```
Selection.Paragraphs.TabStops(InchesToPoints(2)) _
    .Alignment = wdAlignTabRight
```

Use **TabStops(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 document.

```
ActiveDocument.Paragraphs(1).TabStops(1).Clear
```

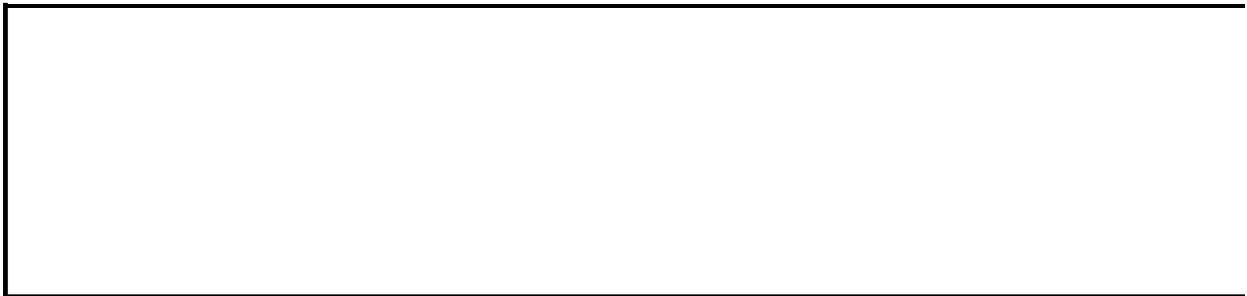
The following example adds a right-aligned tab stop positioned at 2 inches to the selected paragraphs.

```
Selection.Paragraphs.TabStops(InchesToPoints(2)) _  
    .Alignment = wdAlignTabRight
```

Remarks

When working with the **Paragraphs** collection (or a range with several paragraphs), you must modify each paragraph in the collection individually if the tab stops aren't identical in all the paragraphs. The following example removes the tab positioned at 1 inch from every paragraph in the active document.

```
For Each para In ActiveDocument.Content.Paragraphs  
    para.TabStops(InchesToPoints(1)).Clear  
Next para
```



Task Object

[Tasks](#)  [Task](#)

Represents a single task running on the system. The **Task** object is a member of the [Tasks](#) collection. The **Tasks** collection includes all the applications that are currently running on the system.

Using the Task Object

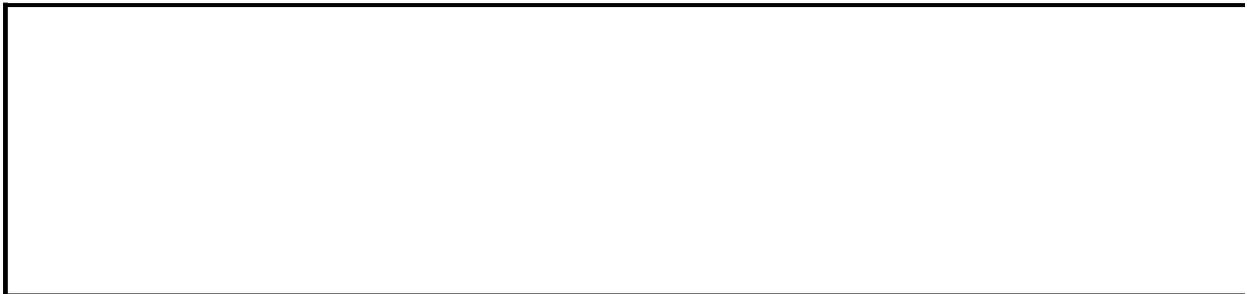
Use **Tasks**(*index*), where *index* is the application name or the index number, to return a single **Task** object. The following example switches to and resizes the application window for the first visible task in the **Tasks** collection.

```
With Tasks(1)
    If .Visible = True Then
        .Activate
        .Width = 400
        .Height = 200
    End If
End With
```

The following example restores the Calculator application window if Calculator is in the **Tasks** collection.

```
If Tasks.Exists("Calculator") = True Then
    Tasks("Calculator").WindowState = wdWindowStateNormal
End If
```

Use Visual Basic's **Shell** function to run an executable program and add the program to the **Tasks** collection.



TaskPane Object

[TaskPanes](#) └ [TaskPane](#)

Represents a single task pane available to Microsoft Word, which contains common tasks that users perform. The **TaskPane** object is a member of the [TaskPanes](#) collection.

Using the TaskPane object

Use the [TaskPanes](#) property to return a **TaskPane** object. Use the [Visible](#) property to display an individual task pane. This example displays the formatting task pane.

```
Sub FormattingPane()  
    Application.TaskPanes(wdTaskPaneFormatting).Visible = True  
End Sub
```



TaskPanes Collection

[Application](#) └ [TaskPanes](#)
└ [TaskPane](#)

A collection of [TaskPane](#) objects that contains commonly performed tasks in Microsoft Word.

Using the TaskPanes collection

Use the [TaskPanes](#) property to return the **TaskPanes** collection. Use the [Item](#) method with a **wdWorkPane** constant to refer to a specific task pane. The example below displays the formatting task pane.

```
Sub FormattingPane()  
    Application.TaskPanes(wdTaskPaneFormatting).Visible = True  
End Sub
```



Tasks Collection Object

Multiple objects [└Tasks](#)
[└Task](#)

A collection of [Task](#) objects that represents all the tasks currently running on the system.

Using the Tasks Collection

Use the [Tasks](#) property to return the **Tasks** collection. The following example determines whether Microsoft Excel is running. If it is, this example switches to it and maximizes it; otherwise, the example starts it.

```
If Tasks.Exists("Microsoft Excel") = True Then
    Tasks("Microsoft Excel").Activate
    Tasks("Microsoft Excel").WindowState = wdWindowStateMaximize
Else
    Shell "C:\Program Files\" & _
        "Microsoft Office\Office10\Excel.exe"
End If
```

Use Visual Basic's **Shell** function to run an executable program and add the program to the **Tasks** collection.

Use **Tasks(index)**, where *index* is the application name or the index number, to return a single **Task** object. The following example opens and resizes the application window for the first visible task in the **Tasks** collection.

```
With Tasks(1)
    If .Visible = True Then
        .Activate
        .Width = 400
        .Height = 200
    End If
End With
```

The following example restores the Calculator application window if the application is in the **Tasks** collection.

```
If Tasks.Exists("Calculator") = True Then
    Tasks("Calculator").WindowState = wdWindowStateNormal
End If
```



Template Object

Multiple objects [└ Templates \(Template\)](#)
└ Multiple objects

Represents a document template. The **Template** object is a member of the [Templates](#) collection. The **Templates** collection includes all the available **Template** objects.

Using the Template Object

Use **Templates**(*index*), where *index* is the template name or the index number, to return a single **Template** object. The following example saves the Memo2.dot template if it's in the **Templates** collection.

```
For Each aTemp In Templates
    If LCase(aTemp.Name) = "memo2.dot" Then aTemp.Save
Next aTemp
```

The index number represents the position of the template in the **Templates** collection. The following example opens the first template in the **Templates** collection.

```
Templates(1).OpenAsDocument
```

The **Add** method isn't available for the **Templates** collection. Instead, you can add a template to the **Templates** collection by doing any of the following:

- Using the [Open](#) method with the **Documents** collection to open a document based on a template or a template
- Using the [Add](#) method with the **Documents** collection to open a new document based on a template
- Using the [Add](#) method with the **Addins** collection to load a global template
- Using the [AttachedTemplate](#) property with the **Document** object to attach a template to a document

Remarks

Use the [NormalTemplate](#) property to return a template object that refers to the Normal template. Use the [AttachedTemplate](#) property to return the template attached to the specified document.

Use the [DefaultFilePath](#) property to return or set the location of user or workgroup templates (that is, the folder where you want to store these templates). The following example displays the user template folder from the **File Locations** tab in the **Options** dialog box (**Tools** menu).

```
MsgBox Options.DefaultFilePath(wdUserTemplatesPath)
```



Templates Collection Object

[Application](#) └ [Templates \(Template\)](#)

└ Multiple objects

A collection of [Template](#) objects that represent all the templates that are currently available. This collection includes open templates, templates attached to open documents, and global templates loaded in the **Templates and Add-ins** dialog box (**Tools** menu).

Using the Templates Collection

Use the **Templates** property to return the **Templates** collection. The following example displays the path and file name of each template in the **Templates** collection.

```
For Each aTemp In Templates
    MsgBox aTemp.FullName
Next aTemp
```

The **Add** method isn't available for the **Templates** collection. Instead, you can add a template to the **Templates** collection by doing any of the following:

- Using the [Open](#) method with the **Documents** collection to open a document based on a template or a template
- Using the [Add](#) method with the **Documents** collection to open a new document based on a template
- Using the [Add](#) method with the **Addins** collection to load a global template
- Using the [AttachedTemplate](#) property with the **Document** object to attach a template to a document

Use **Templates(index)**, where *index* is the template name or the index number, to return a single **Template** object. The following example saves the Dot1.dot template.

```
Templates("C:\MSOffice\WinWord\Templates\Dot1.dot").Save
```

The index number represents the position of the template in the **Templates** collection. The following example displays the file name of the first template in the **Templates** collection.

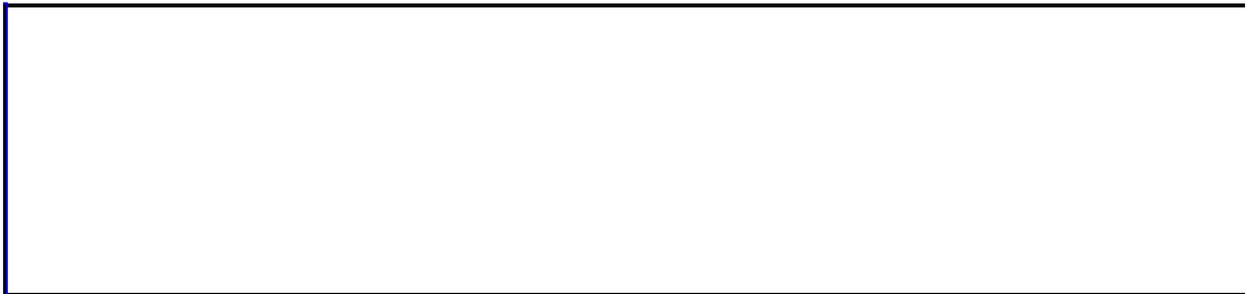
```
MsgBox Templates(1).FullName
```

Remarks

Use the [NormalTemplate](#) property to return a template object that refers to the Normal template. Use the [AttachedTemplate](#) property to return the template attached to the specified document.

Use the [DefaultFilePath](#) property to determine the location of user or workgroup templates (that is, the folder where you want to store these templates). The following example displays the user template folder from the **File Locations** tab in the **Options** dialog box (**Tools** menu).

```
MsgBox Options.DefaultFilePath(wdUserTemplatePath)
```



TextColumn Object

[PageSetup](#)  [TextColumns \(TextColumn\)](#)

Represents a single text column. The **TextColumn** object is a member of the [TextColumns](#) collection. The **TextColumns** collection includes all the columns in a document or section of a document.

Using the TextColumn Object

Use **TextColumns**(*index*), where *index* is the index number, to return a single **TextColumn** object. The index number represents the position of the column in the **TextColumns** collection (counting from left to right).

The following example sets the space after the first text column in the active document to 0.5 inch.

```
ActiveDocument.PageSetup.TextColumns(1).SpaceAfter = _  
    InchesToPoints(0.5)
```

Use the [Add](#) method to add a column to the collection of columns. By default, there's one text column in the **TextColumns** collection. The following example adds a 2.5-inch-wide column to the active document.

```
ActiveDocument.PageSetup.TextColumns.Add _  
    Width:=InchesToPoints(2.5), _  
    Spacing:=InchesToPoints(0.5), EvenlySpaced:=False
```

Remarks

Use the [SetCount](#) method to arrange text into columns. The following example arranges the text in the active document into three columns.

```
ActiveDocument.PageSetup.TextColumns.SetCount NumColumns:=3
```



TextColumns Collection Object

[PageSetup](#) └ [TextColumns \(TextColumn\)](#)

A collection of [TextColumn](#) objects that represent all the columns of text in a document or a section of a document.

Using the TextColumns Collection

Use the **TextColumns** property to return the **TextColumns** collection. The following example formats the columns in the first section in the active document to be evenly spaced, with a line between the columns.

```
With ActiveDocument.Sections(1).PageSetup.TextColumns
    .EvenlySpaced = True
    .LineBetween = True
End With
```

Use the [Add](#) method to add a column to the collection of columns. By default, there's one text column in the **TextColumns** collection. The following example adds a 2.5-inch-wide column to the active document.

```
ActiveDocument.PageSetup.TextColumns.Add _
    Width:=InchesToPoints(2.5), _
    Spacing:=InchesToPoints(0.5), EvenlySpaced:=False
```

Remarks

Use the [SetCount](#) method to arrange text into columns. The following example arranges the text in the active document into three columns.

```
ActiveDocument.PageSetup.TextColumns.SetCount NumColumns:=3
```



TextEffectFormat Object

[Shapes \(Shape\)](#)  [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 active document. For this example to work, shape one must be a WordArt object.

```
With ActiveDocument.Shapes(1).TextEffect
    .FontName = "Courier New"
    .FontBold = True
    .FontItalic = True
End With
```



↳ [Show All](#)

TextFrame Object

Multiple objects [└ TextFrame](#)
[└ Range](#)

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 [Range](#) 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 document.

```
ActiveDocument.Shapes(1).TextFrame.TextRange.Text = "My Text"
```

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 [HasText](#) property to determine whether the text frame contains text, as shown in the following example.

```
For Each s In ActiveDocument.Shapes
    With s.TextFrame
        If .HasText Then MsgBox .TextRange.Text
    End With
Next
```

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 [Next](#) and [Previous](#) 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.

```
Set myTB1 = ActiveDocument.Shapes.AddTextbox _
    (msoTextOrientationHorizontal, 72, 72, 72, 36)
myTB1.TextFrame.TextRange = _
    "This is some text. This is some more text."
Set myTB2 = ActiveDocument.Shapes.AddTextbox _
    (msoTextOrientationHorizontal, 72, 144, 72, 36)
myTB1.TextFrame.Next = myTB2.TextFrame
```

Use the [ContainingRange](#) property to return a [Range](#) object that represents the entire story that flows between linked text frames. The following example checks the spelling of the text in TextBox 3 and of any other text that's linked to

TextBox 3.

```
Set myStory = ActiveDocument.Shapes("TextBox 3") _  
    .TextFrame.ContainingRange  
myStory.CheckSpelling
```



TextInput Object

[FormFields \(FormField\)](#)  [TextInput](#)

Represents a single text form field.

Using the `TextInput` Object

Use `FormFields(index)`, where *index* is either the bookmark name associated with the text form field or the index number, to return a `FormField` object. Use the `TextInput` property with the `FormField` object to return a `TextInput` object. The following example deletes the contents of the text form field named "Text1" in the active document.

```
ActiveDocument.FormFields("Text1").TextInput.Clear
```

The index number represents the position of the form field in the `FormFields` collection. The following example checks the type of the first form field in the active document. If the form field is a text form field, the example sets "Mission Critical" as the value of the field.

```
If ActiveDocument.FormFields(1).Type = wdFieldFormTextInput Then
    ActiveDocument.FormFields(1).Result = "Mission Critical"
End If
```

The following example determines whether the `ffield` variable represents a valid text form field in the active document before it sets the default text.

```
Set ffield = ActiveDocument.FormFields(1).TextInput
If ffield.Valid = True Then
    ffield.Default = "Type your name here"
Else
    MsgBox "First field is not a text box"
End If
```

Use the `Add` method with the `FormFields` object to add a text form field. The following example adds a text form field at the beginning of the active document and then sets the name of the form field to "FirstName."

```
Set ffield = ActiveDocument.FormFields.Add( _
    Range:=ActiveDocument.Range(Start:=0, End:=0), _
    Type:=wdFieldFormTextInput)
ffield.Name = "FirstName"
```



TextRetrievalMode Object

[Range](#)  [TextRetrievalMode](#)

Represents options that control how text is retrieved from a **Range** object.

Using the TextRetrievalMode Object

Use the **TextRetrievalMode** property to return a **TextRetrievalMode** object. The following example displays the text of the first sentence in the active document, excluding field codes and hidden text.

```
With ActiveDocument.Sentences(1).TextRetrievalMode
    .IncludeHiddenText = False
    .IncludeFieldCodes = False
    MsgBox .Parent.Text
End With
```

Remarks

Changing the [ViewType](#), [IncludeHiddenText](#), or [IncludeFieldCodes](#) property of the **TextRetrievalMode** object doesn't change the screen display. Instead, changing one of these properties determines what text is retrieved from a **Range** object when the [Text](#) property is used.

ThreeDFormat Object

[Shapes \(Shape\)](#) └ [ThreeDFormat](#)
└ [ColorFormat](#)

Represents a shape's three-dimensional formatting.

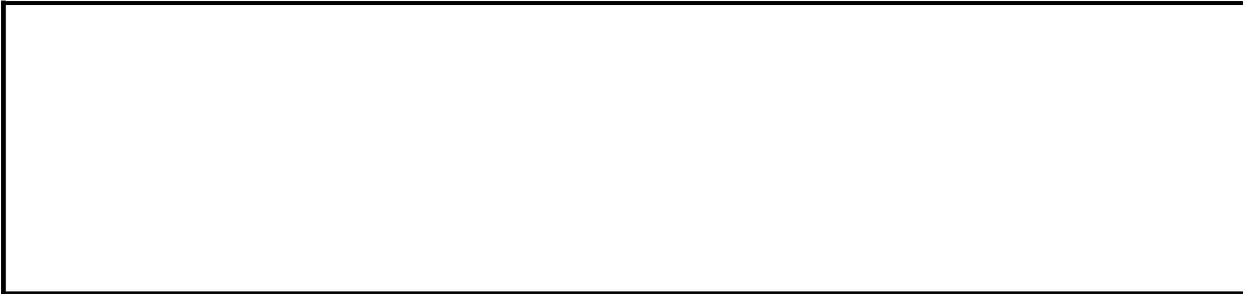
Using The ThreeDFormat Object

Use the **ThreeD** property to return a **ThreeDFormat** object. The following example adds an oval to the active document and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Set myShape = ActiveDocument.Shapes _  
    .AddShape(msoShapeOval, 90, 90, 90, 40)  
With myShape.ThreeD  
    .Visible = True  
    .Depth = 50  
    ' RGB value for purple  
    .ExtrusionColor.RGB = RGB(255, 100, 255)  
End With
```

Remarks

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



TwoInitialCapsException Object

[Application](#) └ [AutoCorrect](#)
└ [TwoInitialCapsExceptions \(TwoInitialCapsException\)](#)

Represents a single initial-capital AutoCorrect exception. The **TwoInitialCapsException** object is a member of the [TwoInitialCapsExceptions](#) collection. The **TwoInitialCapsExceptions** collection includes all the items listed in the **Don't correct** box on the **INitial CAPs** tab in the **AutoCorrect Exceptions** dialog box.

Using the TwoInitialCapsException Object

Use **TwoInitialCapsExceptions**(*index*), where *index* is the initial capital exception name or the index number, to return a single **TwoInitialCapsException** object. The following example deletes the initial-capital exception named "KMenu."

```
AutoCorrect.TwoInitialCapsExceptions("KMenu").Delete
```

The index number represents the position of the initial-capital exception in the **TwoInitialCapsExceptions** collection. The following example displays the name of the first item in the **TwoInitialCapsExceptions** collection.

```
MsgBox AutoCorrect.TwoInitialCapsExceptions(1).Name
```

If the **TwoInitialCapsAutoAdd** property is **True**, words are automatically added to the list of initial-capital exceptions. Use the **Add** method to add an item to the **TwoInitialCapsExceptions** collection. The following example adds "Industry" to the list of initial-capital exceptions.

```
AutoCorrect.TwoInitialCapsExceptions.Add Name:="INDustry"
```



TwoInitialCapsExceptions Collection Object

[Application](#) └ [AutoCorrect](#)
└ [TwoInitialCapsExceptions \(TwoInitialCapsException\)](#)

A collection of [TwoInitialCapsException](#) objects that represent all the items listed in the **Don't correct** box on the **INitial CAPs** tab in the **AutoCorrect Exceptions** dialog box.

Using the **TwoInitialCapsExceptions** Collection

Use the **TwoInitialCapsExceptions** property to return the **TwoInitialCapsExceptions** collection. The following example displays the items in this collection.

```
For Each aCap In AutoCorrect.TwoInitialCapsExceptions
    MsgBox aCap.Name
Next aCap
```

If the **TwoInitialCapsAutoAdd** property is **True**, words are automatically added to the list of initial-capital exceptions. Use the **Add** method to add an item to the **TwoInitialCapsExceptions** collection. The following example adds "Industry" to the list of initial-capital exceptions.

```
AutoCorrect.TwoInitialCapsExceptions.Add Name:="INDustry"
```

Use **TwoInitialCapsExceptions(index)**, where *index* is the initial cap name or the index number, to return a single **TwoInitialCapsException** object. The following example deletes the initial-capital item named "KMenu."

```
AutoCorrect.TwoInitialCapsExceptions("KMenu").Delete
```

The index number represents the position of the initial-capital exception in the **TwoInitialCapsExceptions** collection. The following example displays the name of the first item in the **TwoInitialCapsExceptions** collection.

```
MsgBox AutoCorrect.TwoInitialCapsExceptions(1).Name
```



Variable Object

[Documents \(Document\)](#) └ [Variables \(Variable\)](#)

Represents a variable stored as part of a document. Document variables are used to preserve macro settings in between macro sessions. The **Variable** object is a member of the [Variables](#) collection. The **Variables** collection includes all the document variables in a document or template.

Using the Variable Object

Use **Variables**(*index*), where *index* is the document variable name or the index number, to return a single **Variable** object. The following example displays the value of the Temp document variable in the active document.

```
MsgBox ActiveDocument.Variables("Temp").Value
```

The index number represents the position of the document variable in the **Variables** collection. The last variable added to the **Variables** collection is index number 1; the second-to-last variable added to the collection is index number 2, and so on. The following example displays the name of the first document variable in the active document.

```
MsgBox ActiveDocument.Variables(1).Name
```

Use the [Add](#) method to add a variable to a document. The following example adds a document variable named "Temp" with a value of 12 to the active document.

```
ActiveDocument.Variables.Add Name:="Temp", Value:="12"
```

If you try to add a document variable with a name that already exists in the **Variables** collection, an error occurs. To avoid this error, you can enumerate the collection before adding any new variables. If the Blue document variable already exists in the active document, the following example sets its value to 6. If this variable doesn't already exist, this example adds it to the document and sets it to 6.

```
For Each aVar In ActiveDocument.Variables
    If aVar.Name = "Blue" Then num = aVar.Index
Next aVar
If num = 0 Then
    ActiveDocument.Variables.Add Name:="Blue", Value:=6
Else
    ActiveDocument.Variables(num).Value = 6
End If
```

Remarks

Document variables are invisible to the user unless a DOCVARIABLE field is inserted with the appropriate variable name. The following example adds a document variable named "Temp" to the active document and then inserts a DOCVARIABLE field to display the value in the variable.

```
With ActiveDocument
    .Variables.Add Name:="Temp", Value:="12"
    .Fields.Add Range:=Selection.Range, _
        Type:=wdFieldDocVariable, Text:="Temp"
End With
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
```

To add a document variable to a template, open the template as a document by using the [OpenAsDocument](#) method. The following example stores the user name (from the **Options** dialog box) in the template attached to the active document.

```
ScreenUpdating = False
With ActiveDocument.AttachedTemplate.OpenAsDocument
    .Variables.Add Name:="UserName", Value:=Application.UserName
    .Close SaveChanges:=wdSaveChanges
End With
```



Variables Collection Object

[Documents \(Document\)](#)  [Variables \(Variable\)](#)

A collection of **Variable** objects that represent the variables added to a document or template. Document variables are used to preserve macro settings in between macro sessions.

Using the Variables Collection

Use the **Variables** property to return the **Variables** collection. The following example displays the number of variables in the document named "Sales.doc."

```
MsgBox Documents("Sales.doc").Variables.Count & " variables"
```

Use the [Add](#) method to add a variable to a document. The following example adds a document variable named "Temp" with a value of 12 to the active document.

```
ActiveDocument.Variables.Add Name:="Temp", Value:="12"
```

If you try to add a document variable with a name that already exists in the **Variables** collection, an error occurs. To avoid this error, you can enumerate the collection before adding any new variables. If the Blue document variable already exists in the active document, the following example sets its value to 6. If this variable doesn't already exist, this example adds it to the document and sets it to 6.

```
For Each aVar In ActiveDocument.Variables
    If aVar.Name = "Blue" Then num = aVar.Index
Next aVar
If num = 0 Then
    ActiveDocument.Variables.Add Name:="Blue", Value:=6
Else
    ActiveDocument.Variables(num).Value = 6
End If
```

Use **Variables(index)**, where *index* is the document variable name or the index number, to return a single **Variable** object. The following example displays the value of the Temp document variable in the active document.

```
MsgBox ActiveDocument.Variables("Temp").Value
```

The index number represents the position of the document variable in the **Variables** collection. The first variable added to the **Variables** collection is index number 1; the second variable added to the collection is index number 2, and so on. The following example displays the name of the first document variable in the active document.

```
MsgBox ActiveDocument.Variables(1).Name
```

To add a variable to a template, open the template as a document by using the [OpenAsDocument](#) method. The following example stores the user name (from the **Options** dialog box) in the template attached to the active document.

```
ScreenUpdating = False  
With ActiveDocument.AttachedTemplate.OpenAsDocument  
    .Variables.Add Name:="UserName", Value:= Application.UserName  
    .Close SaveChanges:=wdSaveChanges  
End With
```



Version Object

[Documents \(Document\)](#) └ [Versions \(Version\)](#)

Represents a single version of a document. The **Version** object is a member of the [Versions](#) collection. The **Versions** collection includes all the versions of the specified document.

Using the Version Object

Use **Versions**(*index*), where *index* is the index number, to return a single **Version** object. The index number represents the position of the version in the **Versions** collection. The first version added to the **Versions** collection is index number 1. The following example displays the comment, author, and date of the first version of the active document.

```
If ActiveDocument.Versions.Count >= 1 Then
    With ActiveDocument.Versions(1)
        MsgBox "Comment = " & .Comment & vbCr & "Author = " & _
            .SavedBy & vbCr & "Date = " & .Date
    End With
End If
```

Use the [Save](#) method to add an item to the **Versions** collection. The following example adds a version of the active document with the specified comment.

```
ActiveDocument.Versions.Save _
    Comment:="incorporated Judy's revisions"
```



Versions Collection Object

[Documents \(Document\)](#)  [Versions \(Version\)](#)

A collection of **Version** objects that represent all the versions of a document. Corresponds to the items listed in the **Versions** dialog box (**File** menu).

Using the Versions Collection

Use the **Versions** property to return the **Versions** collection. The following example turns off the option that automatically creates new document versions.

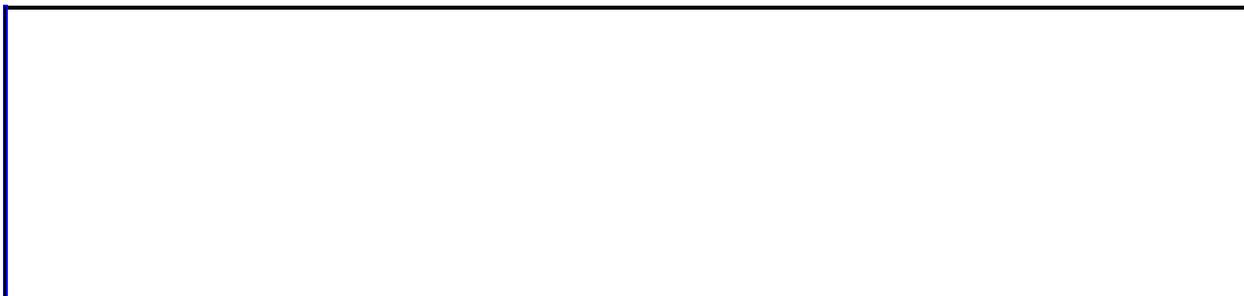
```
ActiveDocument.Versions.AutoVersion = wdAutoVersionOff
```

Use the [Save](#) method to add an item to the **Versions** collection. The following example adds a version with the specified comment.

```
ActiveDocument.Versions.Save _  
    Comment:="incorporated Judy's revisions"
```

Use **Versions(index)**, where *index* is the index number, to return a single **Version** object. The index number represents the position of the version in the **Versions** collection. The first version added to the **Versions** collection is index number 1. The following example displays the comment, author, and date of the first version of the active document.

```
If ActiveDocument.Versions.Count >= 1 Then  
    With ActiveDocument.Versions(1)  
        MsgBox "Comment = " & .Comment & vbCr & "Author = " & _  
            .SavedBy & vbCr & "Date = " & .Date  
    End With  
End If
```



View Object

Multiple objects [↳View](#)
↳Multiple objects

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

Using the View Object

Use the [View](#) property to return the **View** object. The following example sets view options for the active window.

```
With ActiveDocument.ActiveWindow.View
    .ShowAll = True
    .TableGridlines = True
    .WrapToWindow = False
End With
```

Remarks

Use the [Type](#) property to change the view. The following example switches the active window to normal view.

```
ActiveDocument.ActiveWindow.View.Type = wdNormalView
```

Use the [Percentage](#) property to change the size of the text on-screen. The following example enlarges the on-screen text to 120 percent.

```
ActiveDocument.ActiveWindow.View.Zoom.Percentage = 120
```

Use the [SeekView](#) property to view comments, endnotes, footnotes, or the document header or footer. The following example displays the current footer in the active window in print layout view.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdPrintView  
    .SeekView = wdSeekCurrentPageFooter  
End With
```



WebOptions Object

[Documents \(Document\)](#) └ [WebOptions](#)

Contains document-level attributes used by Microsoft Word when you save a document as a Web page or open a Web page. You can return or set attributes either at the application (global) level or at the document level. (Note that attribute values can be different from one document to another, depending on the attribute value at the time the document was saved.) Document-level attribute settings override application-level attribute settings. Application-level attributes are contained in the [DefaultWebOptions](#) object.

Using the WebOptions Object

Use the [WebOptions](#) property to return the **WebOptions** object. The following example checks to see whether PNG (Portable Network Graphics) is allowed as an image format and then sets the strImageFileType variable accordingly.

```
Set objAppWebOptions = ActiveDocument.WebOptions
With objAppWebOptions
    If .AllowPNG = True Then
        strImageFileType = "PNG"
    Else
        strImageFileType = "JPG"
    End If
End With
```



Window Object

Multiple objects [└ Windows \(Window\)](#)
└ Multiple objects

Represents a window. Many document characteristics, such as scroll bars and rulers, are actually properties of the window. The **Window** object is a member of the [Windows](#) collection. The **Windows** collection for the **Application** object contains all the windows in the application, whereas the **Windows** collection for the **Document** object contains only the windows that display the specified document.

Using the Window Object

Use **Windows**(*index*), where *index* is the window name or the index number, to return a single **Window** object. The following example maximizes the Document1 window.

```
Windows("Document1").WindowState = wdWindowStateMaximize
```

The index number is the number to the left of the window name on the **Window** menu. The following example displays the caption of the first window in the **Windows** collection.

```
MsgBox Windows(1).Caption
```

Use the [Add](#) method or the [NewWindow](#) method to add a new window to the **Windows** collection. Each of the following statements creates a new window for the document in the active window.

```
ActiveDocument.ActiveWindow.NewWindow  
NewWindow  
Windows.Add
```

Remarks

A colon (:) and a number appear in the window caption when more than one window is open for a document.

When you switch the view to print preview, a new window is created. This window is removed from the **Windows** collection when you close print preview.



Windows Collection Object

Multiple objects [└ Windows \(Window\)](#)
└ Multiple objects

A collection of [Window](#) objects that represent all the available windows. The **Windows** collection for the **Application** object contains all the windows in the application, whereas the **Windows** collection for the **Document** object contains only the windows that display the specified document.

Using the Windows Collection

Use the **Windows** property to return the **Windows** collection. The following example tiles all the windows so that they don't overlap one another.

```
Windows.Arrange ArrangeStyle:=wdTiled
```

Use the [Add](#) method or the [NewWindow](#) method to add a new window to the **Windows** collection. Each of the following statements creates a new window for the document in the active window.

```
ActiveDocument.ActiveWindow.NewWindow  
NewWindow  
Windows.Add
```

Use **Windows(index)**, where *index* is the window name or the index number, to return a single **Window** object. The following example maximizes the Document1 window.

```
Windows("Document1").WindowState = wdWindowStateMaximize
```

The index number is the number to the left of the window name on the **Window** menu. The following example displays the caption of the first window in the **Windows** collection.

```
MsgBox Windows(1).Caption
```

Remarks

A colon (:) and a number appear in the window caption when more than one window is open for a document.

When you switch the view to print preview, a new window is created. This window is removed from the **Windows** collection when you close print preview.



Words Collection Object

Multiple objects [└ Words](#)
[└ Range](#)

A collection of words in a selection, range, or document. Each item in the **Words** collection is a [Range](#) object that represents one word. There is no Word object.

Using the Words Collection

Use the [Words](#) property to return the **Words** object. The following example displays how many words are currently selected.

```
MsgBox Selection.Words.Count & " words are selected"
```

Use **Words**(*index*), where *index* is the index number, to return a **Range** object that represents one word. The index number represents the position of the word in the **Words** collection. The following example formats the first word in the selection as 24-point italic.

```
With Selection.Words(1)  
    .Italic = True  
    .Font.Size = 24  
End With
```

The item in the **Words** collection includes both the word and the spaces after the word. To remove the trailing spaces, use Visual Basic's **RTrim** function — for example, `RTrim(ActiveDocument.Words(1))`. The following example selects the first word (and its trailing spaces) in the active document.

```
ActiveDocument.Words(1).Select
```

Remarks

If the selection is the insertion point and it is immediately followed by a space, `Selection.Words(1)` refers to the word preceding the selection. If the selection is the insertion point and is immediately followed by a character, `Selection.Words(1)` refers to the word following the selection.

The **Count** property for this collection in a document returns the number of items in the main story only. To count items in other stories use the collection with the **Range** object. Also, the **Count** property includes punctuation and paragraph marks in the total. If you need a count of the the actual words in a document, use the **Word Count** dialog box. The following example retrieves the number of words in the active document and assigns the value to the variable `numWords`.

```
Set temp = Dialogs(wdDialogToolsWordCount)
' Execute the dialog box in order to refresh its data.
temp.Execute
numWords = temp.Words
```

For more information about calling built-in dialog boxes, see [Displaying built-in Word dialog boxes](#).

The **Add** method isn't available for the **Words** collection. Instead, use the **InsertAfter** method or the **InsertBefore** method to add text to a **Range** object. The following example inserts text after the first word in the active document.

```
ActiveDocument.Range.Words(1).InsertAfter "New text "
```



WrapFormat Object

Multiple objects [↳ WrapFormat](#)

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

Using the WrapFormat Object

Use the **WrapFormat** property to return the **WrapFormat** object. The following example adds an oval to the active document and specifies that document 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 document text and the top, bottom, left side, and right side of the square.

```
Set myOval = _
    ActiveDocument.Shapes.AddShape(msoShapeOval, 36, 36, 100, 35)
With myOval.WrapFormat
    .Type = wdWrapSquare
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



Zoom Object

Multiple objects [└Zooms \(Zoom\)](#)

Contains magnification options (for example, the zoom percentage) for a window or pane. The **Zoom** object is a member of the [Zooms](#) collection.

Using the Zoom Object

Use the **Zoom** property of the **View** object to return a single **Zoom** object. The following example sets the zoom percentage for the active window to 110 percent.

```
ActiveDocument.ActiveWindow.View.Zoom.Percentage = 110
```

Use **Zooms**(*index*), where *index* identifies the view type, to return a single **Zoom** object. The view type specified by *index* can be one of the following **WdViewType** constants: **wdMasterView**, **wdNormalView**, **wdOutlineView**, **wdPrintPreview**, **wdPrintView**, or **wdWebView**. The following example sets the magnification for the active window so that an entire page is visible.

```
ActiveDocument.ActiveWindow.ActivePane _  
    .Zooms(wdPrintView).PageFit = wdPageFitFullPage
```

The **Add** method isn't available for the **Zooms** collection. The **Zooms** collection includes a single **Zoom** object for each of the various view types (outline, normal, page layout, and so on).



Zooms Collection Object

[Pane](#) └ [Zooms](#)
└ [Zoom](#)

A collection of [Zoom](#) objects that represents the magnification options for each view (outline, normal, print layout, and so on).

Using the Zooms Collection

Use the [Zooms](#) property to return the **Zooms** collection. The following example sets the zoom percentage for the active window to 100 percent in Normal view.

```
ActiveDocument.ActiveWindow.ActivePane _  
    .Zooms(wdNormalView).Percentage = 100
```

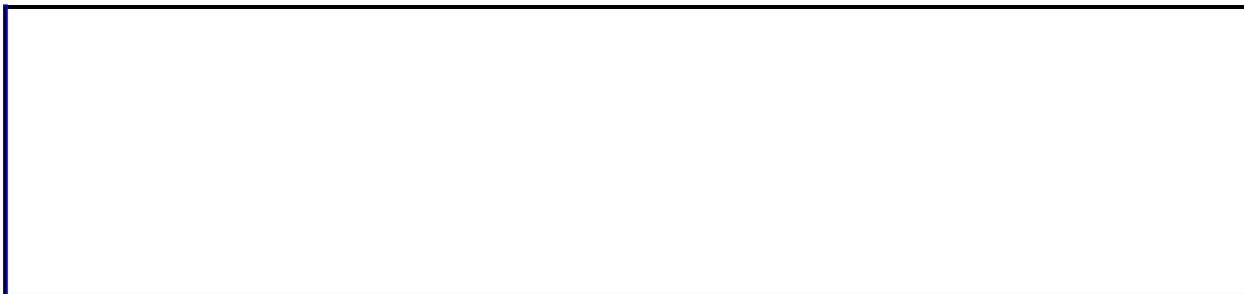
The **Add** method isn't available for the **Zooms** collection. The **Zooms** collection includes a single **Zoom** object for each of the various view types (outline, normal, page layout, and so on). You cannot enumerate the **Zooms** collection by using a **For Each...Next** loop.

Use **Zooms(index)**, where *index* identifies the view type, to return a single **Zoom** object. The view type specified by *index* can be one of the following **WdViewType** constants: **wdMasterView**, **wdNormalView**, **wdOutlineView**, **wdPrintPreview**, **wdPrintView**, or **wdWebView**. The following example sets the magnification for the active window so that an entire page is visible.

```
ActiveDocument.ActiveWindow.ActivePane _  
    .Zooms(wdPrintView).PageFit = wdPageFitFullPage
```

You can also use the [Zoom](#) property of the [View](#) object to return a single **Zoom** object. The following example sets the zoom percentage for the active window to 110 percent.

```
ActiveDocument.ActiveWindow.View.Zoom.Percentage = 110
```



Accept Method

-

Accepts the specified tracked change. The revision marks are removed, and the change is incorporated into the document.

expression.**Accept**

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

Example

This example accepts the next tracked change found if the change type is inserted text.

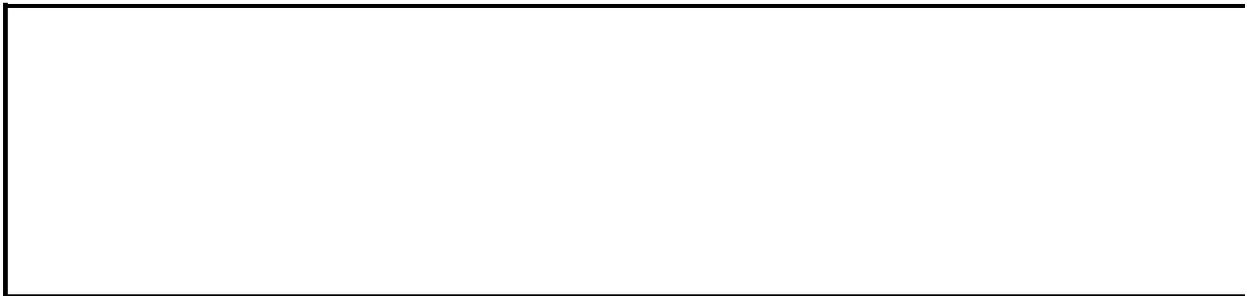
```
Set revNext = Selection.NextRevision(Wrap:=True)

If Not (revNext Is Nothing) Then
    If revNext.Type = wdRevisionInsert Then revNext.Accept
End If
```

This example accepts all the tracked changes in the selection.

```
Dim revLoop As Revision
Dim rngSelection As Range

Set rngSelection = Selection.Range
For Each revLoop In rngSelection.Revisions
    revLoop.Accept
Next revLoop
```



AcceptAll Method

-

Accepts all the tracked changes in a document or range. The revision marks are removed, and the changes are incorporated into the document.

expression.**AcceptAll**

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

Remarks

Use the [AcceptAllRevisions](#) method to accept all revisions in a document.

Example

This example accepts all the tracked changes in the active document.

```
If ActiveDocument.Revisions.Count >= 1 Then _  
    ActiveDocument.Revisions.AcceptAll
```

This example accepts all the tracked changes in the selection.

```
Selection.Range.Revisions.AcceptAll
```



AcceptAllRevisions Method

-
Accepts all tracked changes in the specified document.

expression.**AcceptAllRevisions**

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

Example

This example checks the main story in the active document for tracked changes, and if there are any, the example incorporates all revisions in all stories in the document.

```
If ActiveDocument.Revisions.Count >= 1 Then _  
    ActiveDocument.AcceptAllRevisions
```



AcceptAllRevisionsShown Method

-
Accepts all revisions in the specified document that are displayed on the screen.

expression.**AcceptAllRevisionsShown**

expression Required. An expression that returns a [Document](#) object.

Remarks

Use the [RejectAllRevisionsShown](#) method to reject all revisions in a specified document that are displayed on the screen.

Example

This example accepts all revisions displayed after hiding revisions made by "Jeff Smith." This example assumes that the active document was reviewed by more than one person and that the name of one of the reviewers is "Jeff Smith."

```
Sub AcceptAllChanges()  
    Dim rev As Reviewer  
    With ActiveWindow.View  
        'Display all comments and revisions  
        .ShowRevisionsAndComments = True  
        .ShowFormatChanges = True  
        .ShowInsertionsAndDeletions = True  
  
        For Each rev In .Reviewers  
            rev.Visible = True  
        Next  
  
        'Hide only the revisions/comments made by the  
        'reviewer named "Jeff Smith"  
        .Reviewers(Index:="Jeff Smith").Visible = False  
    End With  
  
    'Accept all revisions displayed in the active view  
    ActiveDocument.AcceptAllRevisionsShown  
  
End Sub
```



↳ [Show All](#)

Activate Method

▸ [Activate method as it applies to the **Application**, **Document**, **InlineShape**, **OLEFormat**, **Pane**, **Shape**, **ShapeRange**, and **Window** objects.](#)

Activates the specified object.

expression.**Activate**

expression Required. An expression that returns one of the above objects.

▸ [Activate method as it applies to the **Task** object.](#)

Activates the **Task** object.

expression.**Activate**(*Wait*)

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

Wait Optional **Variant**. **True** to wait until the user has activated Word before activating the task. **False** to immediately activate the task, even if Word isn't active.

Example

▶ [As it applies to the **Document** object.](#)

This example activates the document named "Sales.doc."

```
Sub OpenSales()  
    'Sales.doc must exist and be open but not active.  
    Documents("Sales.doc").Activate  
End Sub
```

▶ [As it applies to the **Window** object.](#)

This example activates the next window in the **Windows** collection.

```
Sub NextWindow()  
    'Two or more documents must be open for this statement to execut  
    ActiveDocument.ActiveWindow.Next.Activate  
End Sub
```

▶ [As it applies to the **Task** object.](#)

This example activates the Notepad application if Notepad is in the **Tasks** collection.

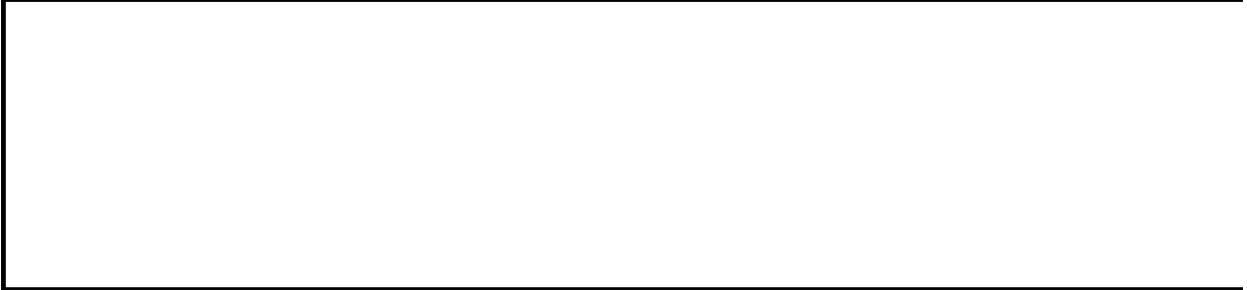
```
Sub ActivateNotePad()  
    Dim Task1    'Notepad must be open and in the Task List.  
  
    For Each Task1 In Tasks  
        If InStr(Task1.Name, "Notepad") > 0 Then  
            Task1.Activate  
            Task1.WindowState = wdWindowStateNormal  
        End If  
    Next Task1  
End Sub
```

▶ [As it applies to the **Pane** object.](#)

This example splits the active window and then activates the first pane.

```
Sub SplitWindow()  
    With ActiveDocument.ActiveWindow
```

```
.SplitVertical = 50  
.Panels(1).Activate  
End With  
End Sub
```



ActivateAs Method

-

Sets the Windows registry value that determines the default application used to activate the specified OLE object.

expression.**ActivateAs**(*ClassType*)

expression Required. An expression that returns an **OLEFormat** object.

ClassType Required **String**. The name of the application in which an OLE object is opened. To see a list of object types that the OLE object can be activated as, click the object and then open the **Convert** dialog box (**Edit** menu, **Object** submenu). You can find the **ClassType** string by inserting an object as an inline shape and then viewing the field codes. The class type of the object follows either the word "EMBED" or the word "LINK."

Example

This example sets the first floating shape on the active document to open in Microsoft Excel, and then it activates the shape. For the example to work, this shape must be an OLE object that can be opened in Microsoft Excel.

```
With ActiveDocument.Shapes(1).OLEFormat  
    .ActivateAs ClassType:="Excel.Sheet"  
    .Activate  
End With
```



↳ [Show All](#)

Add Method

▶ [Add method as it applies to the **AddIns** object.](#)

Returns an **AddIn** object that represents an add-in added to the list of available add-ins.

expression.**Add**(*FileName*, *Install*)

expression Required. An expression that returns an [AddIns](#) object.

FileName Required **String**. The path for the template or WLL.

Install Optional **Variant**. **True** to install the add-in. **False** to add the add-in to the list of add-ins but not install it. The default value is **True**.

Remarks

Use the [Installed](#) property of an add-in to see whether it's already installed.

► [Add method as it applies to the **AutoCorrectEntries** object.](#)

Returns an [AutoCorrectEntry](#) object that represents a plain-text AutoCorrect entry added to the list of available AutoCorrect entries.

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

expression Required. An expression that returns an [AutoCorrectEntries](#) object.

Name Required **String**. The text you want to have automatically replaced with the text specified by **Value**.

Value Required **String**. The text you want to have automatically inserted whenever the text specified by **Name** is typed.

Remarks

Use the [AddRichText](#) method to create a formatted AutoCorrect entry.

▶ [Add method as it applies to the **AutoTextEntries** object.](#)

Returns an [AutoTextEntry](#) object that represents an AutoText entry added to the list of available AutoText entries.

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

expression Required. An expression that returns an [AutoTextEntries](#) object.

Name Required **String**. The text that, when typed, initiates an AutoText entry.

Range Required **Range**. A range of text that will be inserted whenever **Name** is typed.

▶ [Add method as it applies to the **Bookmarks** object.](#)

Returns a [Bookmark](#) object that represents a bookmark added to a range.

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

expression Required. An expression that returns a [Bookmarks](#) object.

Name Required **String**. The name of the bookmark. The name cannot be more than one word.

Range Optional **Variant**. The range of text marked by the bookmark. A bookmark can be set to a collapsed range (the insertion point).

▶ [Add method as it applies to the **CaptionLabels** object.](#)

Returns a [CaptionLabel](#) object that represents a custom caption label.

expression.Add(**Name**)

expression Required. An expression that returns a [CaptionLabels](#) object.

Name Required **String**. The name of the custom caption label.

▶ [Add method as it applies to the **Cells** object.](#)

Returns a **Cell** object that represents a cell added to a table.

expression.Add(BeforeCell)

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

BeforeCell Optional **Variant**. A **Cell** object that represents the cell that will appear immediately to the right of the new cell or cells.

▶ [Add method as it applies to the **Columns** object.](#)

Returns a **Column** object that represents a column added to a table.

expression.Add(BeforeColumn)

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

BeforeColumn Optional **Variant**. A **Column** object that represents the column that will appear immediately to the right of the new column.

▶ [Add method as it applies to the **Comments** object.](#)

Returns a **Comment** object that represents a comment added to a range.

expression.Add(Range, Text)

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

Range Required **Range** object. The range to have a comment added to it.

Text Optional **Variant**. The text of the comment.

▶ [Add method as it applies to the **CustomLabels** object.](#)

Adds a custom mailing label to the **CustomLabels** collection. Returns a **CustomLabel** object that represents the custom mailing label.

expression.Add(Name, DotMatrix)

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

Name Required **String**. The name for the custom mailing labels.

DotMatrix Optional **Variant**. **True** to have the mailing labels printed on a dot-matrix printer.

▶ [Add method as it applies to the **CustomProperties** object.](#)

Returns a [CustomProperty](#) object that represents a custom property added to a smart tag.

expression.Add(Name, Value)

expression Required. An expression that returns a [CustomProperties](#) object.

Name Required **String**. The name of the custom smart tag property.

Value Required **String**. The value of the custom smart tag property

▶ [Add method as it applies to the **Dictionaries** and **HangulHanjaConversionDictionaries** objects.](#)

Returns a [Dictionary](#) object that represents a new custom spelling or conversion dictionary added to the collection of active custom spelling or conversion dictionaries. If a file with the name specified by **FileName** doesn't exist, Microsoft Word creates one.

expression.Add(FileName)

expression Required. An expression that returns one of the above objects.

FileName Required **String**. The string name of the dictionary file. If no path is specified in the string, the proofing tools path is used.

Remarks

The [Dictionaries](#) collection includes only the active custom spelling dictionaries. **Dictionary** objects that are derived from the [Languages](#) collection don't have an **Add** method. These include the **Dictionary** objects returned by the [ActiveSpellingDictionary](#), [ActiveGrammarDictionary](#), [ActiveThesaurusDictionary](#), and [ActiveHyphenationDictionary](#) properties.

Use the [HangulHanjaDictionaries](#) property to return the collection of custom conversion dictionaries. The [HangulHanjaConversionDictionaries](#) collection includes only the active custom conversion dictionaries.

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

► [Add method as it applies to the Documents object](#).

Returns a **Document** object that represents a new, empty document added to the collection of open documents.

expression.Add(Template, NewTemplate, DocumentType, Visible)

expression Required. An expression that returns a [Documents](#) object.

Template Optional **Variant**. The name of the template to be used for the new document. If this argument is omitted, the Normal template is used.

NewTemplate Optional **Variant**. **True** to open the document as a template. The default value is **False**.

DocumentType Optional **Variant**. Can be one of the following **WdNewDocumentType** constants: **wdNewBlankDocument**, **wdNewEmailMessage**, **wdNewFrameset**, or **wdNewWebPage**. The default constant is **wdNewBlankDocument**.

Visible Optional **Variant**. **True** to open the document in a visible window. If this value is **False**, Microsoft Word opens the document but sets the [Visible](#) property of the document window to **False**. The default value is **True**.

▶ [Add method as it applies to the **EmailSignatureEntries** object.](#)

Returns an [EmailSignatureEntry](#) object that represents a new e-mail signature entry.

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

expression Required. An expression that returns an [EmailSignatureEntries](#) object.

Name Required **String**. The name of the e-mail entry.

Range Required **Range** object. The range in the document that will be added as the signature.

Remarks

An e-mail signature is standard text that ends an e-mail message, such as your name and telephone number. Use the [EmailSignatureEntries](#) property to create and manage a collection of e-mail signatures that Microsoft Word will use when creating e-mail messages.

► [Add method as it applies to the Endnotes and Footnotes objects.](#)

Returns an [Endnote](#) or [Footnote](#) object that represents an endnote or footnote added to a range.

expression.Add(Range, Reference, Text)

expression Required. An expression that returns one of the above objects.

Range Required **Range** object. The range marked for the endnote or footnote. This can be a collapsed range.

Reference Optional **Variant**. The text for the custom reference mark. If this argument is omitted, Microsoft Word inserts an automatically-numbered reference mark.

Text Optional **Variant**. The text of the endnote or footnote.

Remarks

To specify a symbol for the **Reference** argument, use the syntax `{FontName CharNum}`. *FontName* is the name of the font that contains the symbol. Names of decorative fonts appear in the **Font** box in the **Symbol** dialog box (**Insert** menu). *CharNum* is the sum of 31 and the number corresponding to the position of the symbol you want to insert, counting from left to right in the table of symbols. For example, to specify an omega symbol (ω) at position 56 in the table of symbols in the Symbol font, the argument would be "{Symbol 87}".

► [Add method as it applies to the **Fields** object.](#)

Adds a **Field** object to the **Fields** collection. Returns the **Field** object at the specified range.

expression.**Add**(*Range*, *Type*, *Text*, *PreserveFormatting*)

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

Range Required **Range** object. The range where you want to add the field. If the range isn't collapsed, the field replaces the range.

Type Optional **Variant**. Can be any **WdFieldType** constant. For a list of valid constants, consult the Object Browser. The default value is **wdFieldEmpty**.

Text Optional **Variant**. Additional text needed for the field. For example, if you want to specify a switch for the field, you would add it here.

PreserveFormatting Optional **Variant**. **True** to have the formatting that's applied to the field preserved during updates.

Remarks

You cannot insert some fields (such as **wdFieldOCX** and **wdFieldFormCheckBox**) by using the **Add** method of the **Fields** collection. Instead, you must use specific methods such as the [AddOLEControl](#) method and the **Add** method for the [FormFields](#) collection.

► [Add method as it applies to the **FirstLetterExceptions**, **OtherCorrectionsExceptions**, and **TwoInitialCapsExceptions** objects.](#)

Returns a [FirstLetterException](#), [OtherCorrectionsExceptions](#), or [TwoInitialCapsExceptions](#) object that represents a new exception added to the list of AutoCorrect exceptions.

expression.**Add**(*Name*)

expression Required. An expression that returns one of the above objects.

Name Required **String**. The word with two initial capital letters that you want Microsoft Word to overlook (**FirstLetterExceptions** object), the abbreviation that you don't want Word to follow with a capital letter (**TwoInitialCapsExceptions** object), or any other word you want Word to overlook (**OtherCorrectionsExceptions** object).

Remarks

If the [TwoInitialCapsAutoAdd](#) property is **True**, words are automatically added to the list of initial-capital exceptions. If the [FirstLetterAutoAdd](#) property is **True**, abbreviations are automatically added to the list of first-letter exceptions. If the [OtherCorrectionsAutoAdd](#) property is **True**, words are automatically added to the list of other corrections exceptions.

► [Add method as it applies to the FormFields object.](#)

Returns a [FormField](#) object that represents a new form field added at a range.

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

expression Required. An expression that returns a [FormFields](#) object.

Range Required **Range** object. The range where you want to add the form field. If the range isn't collapsed, the form field replaces the range.

Type Required [WdFieldType](#). The type of form field to add.

WdFieldType can be one of these WdFieldType constants.

wdFieldAddin

wdFieldAdvance

wdFieldAsk

wdFieldAuthor

wdFieldAutoNum

wdFieldAutoNumLegal

wdFieldAutoNumOutline

wdFieldAutoText

wdFieldAutoTextList

wdFieldBarCode

wdFieldComments

wdFieldCompare

wdFieldCreateDate

wdFieldData
wdFieldDatabase
wdFieldDate
wdFieldDDE
wdFieldDDEAuto
wdFieldDocProperty
wdFieldDocVariable
wdFieldEditTime
wdFieldEmbed
wdFieldEmpty
wdFieldExpression
wdFieldFileName
wdFieldFileSize
wdFieldFillIn
wdFieldFootnoteRef
wdFieldFormCheckBox
wdFieldFormDropDown
wdFieldFormTextInput
wdFieldFormula
wdFieldGlossary
wdFieldGoToButton
wdFieldHTMLActiveX
wdFieldHyperlink
wdFieldIf
wdFieldImport
wdFieldInclude
wdFieldIncludePicture
wdFieldIncludeText
wdFieldIndex
wdFieldIndexEntry
wdFieldInfo
wdFieldKeyWord
wdFieldLastSavedBy

wdFieldLink
wdFieldListNum
wdFieldMacroButton
wdFieldMergeField
wdFieldMergeRec
wdFieldMergeSeq
wdFieldNext
wdFieldNextIf
wdFieldNoteRef
wdFieldNumChars
wdFieldNumPages
wdFieldNumWords
wdFieldOCX
wdFieldPage
wdFieldPageRef
wdFieldPrint
wdFieldPrintDate
wdFieldPrivate
wdFieldQuote
wdFieldRef
wdFieldRefDoc
wdFieldRevisionNum
wdFieldSaveDate
wdFieldSection
wdFieldSectionPages
wdFieldSequence
wdFieldSet
wdFieldSkipIf
wdFieldStyleRef
wdFieldSubject
wdFieldSubscriber
wdFieldSymbol
wdFieldTemplate

wdFieldTime
wdFieldTitle
wdFieldTOA
wdFieldTOAEntry
wdFieldTOC
wdFieldTOCEntry
wdFieldUserAddress
wdFieldUserInitials
wdFieldUserName

▶ [Add method as it applies to the **Frames** object.](#)

Returns a **Frame** object that represents a new frame added to a range, selection, or document.

expression.**Add**(**Range**)

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

Range Required **Range** object. The range that you want the frame to surround.

▶ [Add method as it applies to the **HangulAndAlphabetExceptions** object.](#)

Returns a **HangulAndAlphabetException** object that represents a new exception to the list of AutoCorrect exceptions.

expression.**Add**(**Name**)

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

Name Required **String**. The word that you don't want Microsoft Word to correct automatically.

Remarks

If the [HangulAndAlphabetAutoAdd](#) property is set to **True**, words are automatically added to the list of hangul and alphabet AutoCorrect exceptions.

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

▶ [Add method as it applies to the **HeadingStyles** object.](#)

Returns a [HeadingStyle](#) object that represents a new heading style added to a document. The new heading style will be included whenever you compile a table of contents or table of figures.

expression.**Add**(*Style*, *Level*)

expression Required. An expression that returns a [HeadingStyles](#) object.

Style Required **Variant**. The style you want to add. You can specify this argument by using either the string name for the style or a [Style](#) object.

Level Required **Integer**. A number that represents the level of the heading.

▶ [Add method as it applies to the **HTMLDivisions** object.](#)

Returns an [HTMLDivision](#) object that represents a new HTML division added to a Web document.

expression.**Add**(*Range*)

expression Required. An expression that returns an [HTMLDivisions](#) object.

Range Optional **Variant**. An existing HTML division around which to place the new HTML division.

▶ [Add method as it applies to the **Hyperlinks** object.](#)

Returns a [Hyperlink](#) object that represents a new hyperlink added to a range, selection, or document.

expression.Add(Anchor, Address, SubAddress, ScreenTip, TextToDisplay, Target)

expression Required. An expression that returns a [Hyperlinks](#) object.

Anchor Required **Object**. The text or graphic that you want turned into a hyperlink.

Address Optional **Variants**. The address for the specified link. The address can be an e-mail address, an Internet address, or a file name. Note that Microsoft Word doesn't check the accuracy of the address.

SubAddress Optional **Variants**. The name of a location within the destination file, such as a bookmark, named range, or slide number.

ScreenTip Optional **Variants**. The text that appears as a ScreenTip when the mouse pointer is positioned over the specified hyperlink. The default value is **Address**.

TextToDisplay Optional **Variants**. The display text of the specified hyperlink. The value of this argument replaces the text or graphic specified by **Anchor**.

Target Optional **Variants**. The name of the frame or window in which you want to load the specified hyperlink.

► [Add method as it applies to the Indexes object.](#)

Returns an [Index](#) object that represents a new index added to a document.

expression.Add(Range, HeadingSeparator, RightAlignPageNumbers, Type, NumberOfColumns, AccentedLetters, SortBy, IndexLanguage)

expression Required. An expression that returns an [Indexes](#) object.

Range Required **Range** object. The range where you want the index to appear. The index replaces the range, if the range isn't collapsed.

HeadingSeparator Optional **Variants**. The text between alphabetic groups (entries that start with the same letter) in the index. Can be one of the following **WdHeadingSeparator** constants: **wdHeadingSeparatorBlankLine**,

wdHeadingSeparatorLetter, **wdHeadingSeparatorLetterFull**, **wdHeadingSeparatorLetterLow**, or **wdHeadingSeparatorNone**.

RightAlignPageNumbers Optional **Variant**. **True** to align page numbers with the right margin.

Type Optional **Variant**. Specifies whether subentries are on the same line (run-in) as the main entry or on a separate line (indented) from the main entry. Can be either of the following **WdIndexType** constants: **wdIndexIndent** or **wdIndexRunin**.

NumberOfColumns Optional **Variant**. The number of columns for each page of the index. Specifying 0 (zero) sets the number of columns in the index to the same number as in the document.

AccentedLetters Optional **Variant**. **True** to include separate headings for accented letters in the index (for example, words that begin with "À" and words that begin with "A" are listed under separate headings).

SortBy Optional **Variant**. The sorting criteria to be used for the specified index. Can be either of the following **WdIndexSortBy** constants: **wdIndexSortByStroke** or **wdIndexSortBySyllable**.

IndexLanguage Optional **Variant**. The sorting language to be used for the specified index. Can be any of the **WdLanguageID** constants. For the list of valid **WdLanguageID** constants, see the Object Browser in the Visual Basic Editor.

Remarks

An index is built from Index Entry (XE) fields in a document. Use the [MarkEntry](#) method to mark index entries to be included in an index.

► [Add method as it applies to the KeyBindings object.](#)

Returns a [KeyBinding](#) object that represents a new shortcut key for a macro, built-in command, font, AutoText entry, style, or symbol.

expression.Add(**KeyCategory**, **Command**, **KeyCode**, **KeyCode2**, **CommandParameter**)

expression Required. An expression that returns a [KeyBindings](#) object.

KeyCategory Required [WdKeyCategory](#). The category of the key assignment.

WdKeyCategory can be one of these WdKeyCategory constants.

wdKeyCategoryAutoText

wdKeyCategoryCommand

wdKeyCategoryDisable

wdKeyCategoryFont

wdKeyCategoryMacro

wdKeyCategoryNil

wdKeyCategoryPrefix

wdKeyCategoryStyle

wdKeyCategorySymbol

Command Required **String**. The command that the specified key combination executes.

KeyCode Required **Long**. A key you specify by using one of the **WdKey** constants.

KeyCode2 Optional **VARIANT**. A second key you specify by using one of the **WdKey** constants.

CommandParameter Optional **Variant**. Additional text, if any, required for the command specified by ***Command***. For details, see the Remarks section below.

Remarks

You can use the [BuildKeyCode](#) method to create the *KeyCode* or *KeyCode2* argument.

In the following table, the left-hand column contains commands that require a command value, and the right-hand column describes what you must do to specify *CommandParameter* for each of these commands. (The equivalent action in the **Customize Keyboard** dialog box (**Tools** menu) to specifying *CommandParameter* is selecting an item in the list box that appears when you select one of the following commands in the **Commands** box.)

If Command is set to	CommandParameter must be
Borders, Color, or Shading	A number — specified as text — corresponding to the position of the setting selected in the list box that contains values, where 0 (zero) is the first item, 1 is the second item, and so on
Columns	A number between 1 and 45 — specified as text — corresponding to the number of columns you want to apply
Condensed	A text measurement between 0.1 point and 12.75 points specified in 0.05-point increments (72 points = 1 inch)
Expanded	A text measurement between 0.1 point and 12.75 points specified in 0.05-point increments (72 points = 1 inch)
FileOpenFile	The path and file name of the file to be opened. If the path isn't specified, the current folder is used.
Font Size	A positive text measurement, specified in 0.5-point increments (72 points = 1 inch)
Lowered, Raised	A text measurement between 1 point and 64 points, specified in 0.5-point increments (72 points = 1 inch)
Symbol	A string created by concatenating a Chr() instruction and the name of a symbol font (for example, Chr(167) & "Symbol")

► [Add method as it applies to the ListEntries object.](#)

Returns a [ListEntry](#) object that represents an item added to a drop-down form field.

expression.Add(Name, Index)

expression Required. An expression that returns a [ListEntries](#) object.

Name Required **String**. The name of the drop-down form field item.

Index Optional **Variant**. A number that represents the position of the item in the list.

► [Add method as it applies to the ListTemplates object.](#)

Returns a [ListTemplate](#) object that represents a new list template.

expression.Add(OutlineNumbered, Name)

expression Required. An expression that returns a [ListTemplates](#) object.

OutlineNumbered Optional **Variant**. **True** to apply outline numbering to the new list template.

Name Optional **Variant**. An optional name used for linking the list template to a LISTNUM field. You cannot use this name to index the list template in the collection.

Remarks

You cannot use the **Add** method on **ListTemplates** objects returned from a [ListGallery](#) object. You can, however, modify the existing list templates in the galleries.

► [Add method as it applies to the MailMergeFields object.](#)

Returns a [MailMergeField](#) object that represents a mail merge field added to the data source document.

expression.Add(Range, Name)

expression Required. An expression that returns a [MailMergeFields](#) object.

Range Required **Range** object. The range where you want the field to appear. This field replaces the range, if the range isn't collapsed.

Name Required **String**. The name of the field.

► [Add method as it applies to the PageNumbers object.](#)

Returns a [PageNumber](#) object that represents page numbers added to a header or footer in a section.

expression.Add(PageNumberAlignment, FirstPage)

expression Required. An expression that returns a [PageNumbers](#) object.

PageNumberAlignment Optional **Variant**. Can be any [WdPageNumberAlignment](#) constant.

wdAlignPageNumberCenter
wdAlignPageNumberInside
wdAlignPageNumberLeft
wdAlignPageNumberOutside
wdAlignPageNumberRight

FirstPage Optional **Variant. False** to make the first-page header and the first-page footer different from the headers and footers on all subsequent pages in the document. If ***FirstPage*** is set to **False**, a page number isn't added to the first page. If this argument is omitted, the setting is controlled by the [**DifferentFirstPageHeaderFooter**](#) property.

Remarks

If the [LinkToPrevious](#) property for the [HeaderFooter](#) object is set to **True**, the page numbers will continue sequentially from one section to next throughout the document.

► [Add method as it applies to the Panes object.](#)

Returns a [Pane](#) object that represents a new pane to a window.

expression.**Add**(*SplitVertical*)

expression Required. An expression that returns a [Panes](#) object.

SplitVertical Optional **Variant**. A number that represents the percentage of the window, from top to bottom, you want to appear above the split.

Remarks

This method will fail if it's applied to a window that's already been split.

► [Add method as it applies to the **Paragraphs** object.](#)

Returns a **Paragraph** object that represents a new, blank paragraph added to a document.

expression.**Add**(*Range*)

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

Range Optional **Variant**. The range before which you want the new paragraph to be added. The new paragraph doesn't replace the range.

Remarks

If **Range** isn't specified, the new paragraph is added after the selection or range or at the end of the document, depending on *expression*.

▶ [Add method as it applies to the **RecentFiles** object.](#)

Returns a [RecentFile](#) object that represents a file added to the list of recently used files.

expression.**Add(Document, ReadOnly)**

expression Required. An expression that returns a [RecentFile](#) object.

Document Required **Variant**. The document you want to add to the list of recently used files. You can specify this argument by using either the string name for the document or a [Document](#) object.

ReadOnly Optional **Variant**. **True** to make the document read-only.

▶ [Add method as it applies to the **Rows** object.](#)

Returns a [Row](#) object that represents a row added to a table.

expression.**Add(BeforeRow)**

expression Required. An expression that returns a [Rows](#) object.

BeforeRow Optional **Variant**. A **Row** object that represents the row that will appear immediately below the new row.

▶ [Add method as it applies to the **Sections** object.](#)

Returns a [Section](#) object that represents a new section added to a document.

expression.**Add(Range, Start)**

expression Required. An expression that returns a [Sections](#) object.

Range Optional **Variant**. The range before which you want to insert the section break. If this argument is omitted, the section break is inserted at the end of the document.

Start Optional **Variant**. The type of section break you want to add. Can be one of the following **WdSectionStart** constants: **wdSectionContinuous**, **wdSectionEvenPage**, **wdSectionNewColumn**, **wdSectionNewPage**, or **wdSectionOddPage**. If this argument is omitted, a Next Page section break is added.

▶ [Add method as it applies to the **SmartTags** object.](#)

Returns a **SmartTag** object that represents a new smart tag added to a document.

expression.Add(Name, Range, Properties)

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

Name Required **String**. The name of the smart tag.

Range Optional **Variant**. The range to which to apply the smart tag.

Properties Optional **Variant**. Properties that apply to the smart tag.

▶ [Add method as it applies to the **Styles** object.](#)

Returns a **Style** object that represents a new user-defined style added to the list of styles.

expression.Add(Name, Type)

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

Name Required **String**. The string name for the new style.

Type Optional **Variant**. The style type of the new style. Can be one of the following **WdStyleType** constants: **wdStyleTypeParagraph**, **wdStyleTypeCharacter**, **wdStyleTypeList**, or **wdStyleTypeTable**.

▶ [Add method as it applies to the **StyleSheets** object.](#)

Returns a [StyleSheet](#) object that represents a new style sheet added to a Web document.

expression.Add(FileName, LinkType, Title, Precedence)

expression Required. An expression that returns a [StyleSheets](#) object.

FileName Required **String**. The path and file name of the cascading style sheet.

LinkType Optional [WdStyleSheetLinkType](#). Indicates whether the style sheet should be added as a link or imported into the Web document.

WdStyleSheetLinkType can be one of these WdStyleSheetLinkType constants.

wdStyleSheetLinkTypeImported

wdStyleSheetLinkTypeLinked *default*

Title Optional **String**. The name of the style sheet.

Precedence Optional [WdStyleSheetPrecedence](#). Indicates the level of importance compared to other cascading style sheets attached to the Web document.

WdStyleSheetPrecedence can be one of these WdStyleSheetPrecedence constants.

wdStyleSheetPrecedenceHigher

wdStyleSheetPrecedenceHighest *default*

wdStyleSheetPrecedenceLower

wdStyleSheetPrecedenceLowest

► [Add method as it applies to the Tables object.](#)

Returns a [Table](#) object that represents a new, blank table added to a document.

expression.Add(Range, NumRows, NumColumns, DefaultTableBehavior, AutoFitBehavior)

expression Required. An expression that returns a [Tables](#) object.

Range Required **Range** object. The range where you want the table to appear.

The table replaces the range, if the range isn't collapsed.

NumRows Required **Long**. The number of rows you want to include in the table.

NumColumns Required **Long**. The number of columns you want to include in the table.

DefaultTableBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word automatically resizes cells in tables to fit the cells' contents (AutoFit). Can be either of the following constants: **wdWord8TableBehavior** (AutoFit disabled) or **wdWord9TableBehavior** (AutoFit enabled). The default constant is **wdWord8TableBehavior**.

AutoFitBehavior Optional **Variant**. Sets the AutoFit rules for how Word sizes tables. Can be one of the following **WdAutoFitBehavior** constants: **wdAutoFitContent**, **wdAutoFitFixed**, or **wdAutoFitWindow**. If **DefaultTableBehavior** is set to **wdWord8TableBehavior**, this argument is ignored.

► [Add method as it applies to the **TablesOfAuthorities** object.](#)

Returns a [TableOfAuthorities](#) object that represents a table of authorities added to a document.

expression.Add(Range, Category, Bookmark, Passim, KeepEntryFormatting, Separator, IncludeSequenceName, EntrySeparator, PageRangeSeparator, IncludeCategoryHeader, PageNumberSeparator)

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

Range Required **Range** object. The range where you want the table of authorities to appear. The table of authorities replaces the range, if the range isn't collapsed.

Category Optional **Variant**. The category of entries you want to include in the table of authorities. Corresponds to the \c switch for a Table of Authorities (TOA) field. Values 0 through 16 correspond to the items listed in the **Category** box on the **Table of Authorities** tab in the **Index and Tables** dialog box (**Reference** command, **Insert** menu). The default value is 1.

Bookmark Optional **VARIANT**. The string name of the bookmark from which you want to collect entries for the table of authorities. If **Bookmark** is specified, the entries are collected only from the portion of the document marked by the bookmark. Corresponds to the \b switch for a Table of Authorities (TOA) field.

Passim Optional **VARIANT**. **True** to replace five or more page references to the same authority with Passim in the table of authorities. Corresponds to the \p switch for a Table of Authorities (TOA) field. If this argument is omitted, **Passim** is assumed to be **False**.

KeepEntryFormatting Optional **VARIANT**. **True** to apply formatting from table of authorities entries to the entries in the table of authorities. Corresponds to the \f switch for a Table of Authorities (TOA) field. If this argument is omitted, **KeepEntryFormatting** is assumed to be **True**.

Separator Optional **VARIANT**. The characters (up to five) between each sequence number and its page number in the table of authorities. Corresponds to the \d switch for a Table of Authorities (TOA) field. If argument is omitted, a hyphen (-) is used.

IncludeSequenceName Optional **VARIANT**. A string that specifies the Sequence (SEQ) field identifier for the table of authorities. Corresponds to the \s switch for a Table of Authorities (TOA) field.

EntrySeparator Optional **VARIANT**. The characters (up to five) that separate each entry and its page number in the table of authorities. Corresponds to the \e switch for a Table of Authorities (TOA) field. If this argument is omitted, no separator is used.

PageRangeSeparator Optional **VARIANT**. The characters (up to five) that separate the beginning and ending page numbers in each page range the table of authorities. Corresponds to the \g switch for a Table of Authorities (TOA) field. If this argument is omitted, an en dash is used.

IncludeCategoryHeader Optional **VARIANT**. **True** to have the category name for each group of entries appear in the table of authorities (for example, Cases). Corresponds to the \h switch for a Table of Authorities (TOA) field. If this argument is omitted, **IncludeCategoryHeader** is assumed to be **True**.

PageNumberSeparator Optional **VARIANT**. The characters (up to five) that

separate individual page numbers within page references in the table of authorities. Corresponds to the \l switch for a Table of Authorities (TOA) field. If this argument is omitted, a comma and a space are used.

Remarks

A table of authorities is built from Table of Authorities Entry (TA) fields in a document. Use the [MarkCitation](#) method to mark citations to be included in the table of authorities.

► [Add method as it applies to the TablesOfContents object.](#)

Returns a [TableOfContents](#) object that represents a table of contents added to a document.

expression.Add(Range, UseHeadingStyles, UpperHeadingLevel, LowerHeadingLevel, UseFields, TableID, RightAlignPageNumbers, IncludePageNumbers, AddedStyles, UseHyperlinks, HidePageNumbersInWeb, UseOutlineLevels)

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

Range Required **Range** object. The range where you want the table of contents to appear. The table of contents replaces the range, if the range isn't collapsed.

UseHeadingStyles Optional **VARIANT**. **True** to use built-in heading styles to create the table of contents. The default value is **True**.

UpperHeadingLevel Optional **VARIANT**. The starting heading level for the table of contents. Corresponds to the starting value used with the \o switch for a Table of Contents (TOC) field. The default value is 1.

LowerHeadingLevel Optional **VARIANT**. The ending heading level for the table of contents. Corresponds to the ending value used with the \o switch for a Table of Contents (TOC) field. The default value is 9.

UseFields Optional **VARIANT**. **True** if Table of Contents Entry (TC) fields are used to create the table of contents. Use the [MarkEntry](#) method to mark entries to be included in the table of contents. The default value is **False**.

TableID Optional **VARIANT**. A one-letter identifier that's used to build a table of contents from TC fields. Corresponds to the \f switch for a Table of Contents

(TOC) field. For example, "T" builds a table of contents from TC fields using the table identifier T. If this argument is omitted, TC fields aren't used.

RightAlignPageNumbers Optional **Variant**. **True** if page numbers in the table of contents are aligned with the right margin. The default value is **True**.

IncludePageNumbers Optional **Variant**. **True** to include page numbers in the table of contents. The default value is **True**.

AddedStyles Optional **Variant**. The string name for additional styles used to compile the table of contents (styles other than the Heading 1 – Heading 9 styles). Use the **Add** method of a [HeadingStyles](#) object to create new heading styles.

UseHyperlinks Optional **Variant**. **True** if entries in a table of contents should be formatted as hyperlinks when the document is being publishing to the Web. The default value is **True**.

HidePageNumbersInWeb Optional **Variant**. **True** if page numbers in a table of contents should be hidden when the document is being publishing to the Web. The default value is **True**.

UseOutlineLevels Optional **Variant**. **True** to use outline levels to create the table of contents. The default is **False**.

► [Add method as it applies to the TablesOfFigures object.](#)

Returns a [TableOfFigures](#) object that represents a table of figures added to a document.

expression.Add(Range, Caption, IncludeLabel, UseHeadingStyles, UpperHeadingLevel, LowerHeadingLevel, UseFields, TableID, RightAlignPageNumbers, IncludePageNumbers, AddedStyles, UseHyperlinks, HidePageNumbersInWeb)

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

Range Required **Range** object. The range where you want the table of figures to appear.

Caption Optional **Variant**. The label that identifies the items you want to include in the table of figures. Corresponds to the \c switch for a Table of Contents (TOC) field. The default value is "Figure."

IncludeLabel Optional **Variant**. **True** to include the caption label and caption number in the table of figures. The default value is **True**.

UseHeadingStyles Optional **Variant**. **True** to use built-in heading styles to create the table of figures. The default value is **False**.

UpperHeadingLevel Optional **Variant**. The starting heading level for the table of figures, if **UseHeadingStyles** is set to **True**. Corresponds to the starting value used with the \o switch for a Table of Contents (TOC) field. The default value is 1.

LowerHeadingLevel Optional **Variant**. The ending heading level for the table of figures, if **UseHeadingStyles** is set to **True**. Corresponds to the ending value used with the \o switch for a Table of Contents (TOC) field. The default value is 9.

UseFields Optional **Variant**. **True** to use Table of Contents Entry (TC) fields to create the table of figures. Use the [MarkEntry](#) method to mark entries you want to include in the table of figures. The default value is **False**.

TableID Optional **Variant**. A one-letter identifier that's used to build a table of figures from Table of Contents Entry (TC) fields. Corresponds to the \f switch for a Table of Contents (TOC) field. For example, "i" builds a table of figures for an illustration.

RightAlignPageNumbers Optional **Variant**. **True** align page numbers with the right margin in the table of figures. The default value is **True**.

IncludePageNumbers Optional **Variant**. **True** if page numbers are included in the table of figures. The default value is **True**.

AddedStyles Optional **Variant**. The string name for additional styles used to compile the table of figures (styles other than the Heading 1 – Heading 9 styles).

UseHyperlinks Optional **Variant**. **True** if entries in a table of figures should be formatted as hyperlinks when publishing to the Web. The default value is **True**.

HidePageNumbersInWeb Optional **Vari**ant. **True** if page numbers in a table of figures should be hidden when publishing to the Web. The default value is **True**.

▶ [Add method as it applies to the **TabStops** object.](#)

Returns a **TabStop** object that represents a custom tab stop added to a document.

expression.Add(Position, Alignment, Leader)

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

Position Required **Single**. The position of the tab stop (in points) relative to the left margin.

Alignment Optional **Vari**ant. The alignment of the tab stop. Can be one of the following **WdTabAlignment** constants: **wdAlignTabBar**, **wdAlignTabCenter**, **wdAlignTabDecimal**, **wdAlignTabLeft**, **wdAlignTabList**, or **wdAlignTabRight**. If this argument is omitted, **wdAlignTabLeft** is used.

Leader Optional **Vari**ant. The type of leader for the tab stop. Can be one of the following **WdTabLeader** constants: **wdTabLeaderDashes**, **wdTabLeaderDots**, **wdTabLeaderHeavy**, **wdTabLeaderLines**, **wdTabLeaderMiddleDot**, or **wdTabLeaderSpaces**. If this argument is omitted, **wdTabLeaderSpaces** is used.

▶ [Add method as it applies to the **TextColumns** object.](#)

Returns a **TextColumn** object that represents a new text column added to a section or document.

expression.Add(Width, Spacing, EvenlySpaced)

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

Width Optional **Vari**ant. The width of the new text column in the document, in points.

Spacing Optional **Vari**ant. The spacing between the text columns in the document, in points.

EvenlySpaced Optional **Variant**. **True** to evenly space all the text columns be in the document.

▶ [Add method as it applies to the **Variables** object.](#)

Returns a **Variable** object that represents a variable added to a document.

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

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

Name Required **String**. The name of the document variable.

Value Optional **Variant**. The value for the document variable.

Remarks

Document variables are invisible to the user unless a DOCVARIABLE field is inserted with the appropriate variable name. If you try to add a variable with a name that already exists in the **Variables** collection, an error occurs. To avoid this error, you can enumerate the collection before adding a new variable to it.

▶ [Add method as it applies to the **Windows** object.](#)

Returns a [Window](#) object that represents a new window of a document.

expression.**Add**(*Window*)

expression Required. An expression that returns a [Windows](#) object.

Window Optional **Variant**. The **Window** object you want to open another window for. If this argument is omitted, a new window is opened for the active document.

Remarks

A colon (:) and a number appear in the window caption when more than one window is open for the document.

Example

▶ [As it applies to the **AddIns** object.](#)

This example installs a template named MyFax.dot and adds it to the list of add-ins in the **Templates and Add-ins** dialog box.

```
Sub AddTemplate()  
    ' For this example to work correctly, verify that the  
    ' path is correct and the file exists.  
  
    AddIns.Add FileName:="C:\Program Files\Microsoft Office" _  
        & "\Templates\Letters & Faxes\MyFax.dot", Install:=True  
End Sub
```

▶ [As it applies to the **AutoCorrectEntries** object.](#)

This example adds a plain-text AutoCorrect entry for a common misspelling of the word their.

```
AutoCorrect.Entries.Add Name:="thier", Value:="their"
```

▶ [As it applies to the **AutoTextEntries** object.](#)

This example adds an AutoText entry named Sample Text that contains the text in the selection. This example assumes you have text selected in the active document.

```
Sub AutoTtxt()  
    NormalTemplate.AutoTextEntries.Add Name:="Sample Text", _  
        Range:=Selection.Range  
End Sub
```

▶ [As it applies to the **Bookmarks** object.](#)

This example adds a bookmark named myplace to the selection in the active document.

```
Sub BMark()  
    ' Select some text in the active document prior  
    ' to execution.
```

```
ActiveDocument.Bookmarks.Add _  
    Name:="myplace", Range:=Selection.Range  
End Sub
```

This example adds a bookmark named mark at the insertion point.

```
Sub Mark()  
    ActiveDocument.Bookmarks.Add Name:="mark"  
End Sub
```

This example adds a bookmark named third_para to the third paragraph in Letter.doc, and then it displays all the bookmarks for the document in the active window.

```
Sub ThirdPara()  
    Dim myDoc As Document  
  
    ' To best illustrate this example,  
    ' Letter.doc must be opened, not active,  
    ' and contain more than 3 paragraphs.  
    Set myDoc = Documents("Letter.doc")  
    myDoc.Bookmarks.Add Name:="third_para", _  
        Range:=myDoc.Paragraphs(3).Range  
    myDoc.ActiveWindow.View.ShowBookmarks = True  
End Sub
```

▶ [As it applies to the **CaptionLabels** object.](#)

This example adds a custom caption label named Demo Slide. To verify that the custom label is added, view the **Label** combo box in the **Caption** dialog box, accessed from the **Reference** item on the **Insert** menu.

```
Sub CapLbl()  
    CaptionLabels.Add Name:="Demo Slide"  
End Sub
```

▶ [As it applies to the **Columns** object.](#)

This example creates a table with two columns and two rows in the active document and then adds another column before the first column. The width of the new column is set at 1.5 inches.

```
Sub AddATable()  
    Dim myTable As Table
```

```

Dim newCol As Column

Set myTable = ActiveDocument.Tables.Add(Selection.Range, 2, 2)
Set newCol = myTable.Columns.Add(BeforeColumn:=myTable.Columns(1)
newCol.SetWidth ColumnWidth:=InchesToPoints(1.5), _
    RulerStyle:=wdAdjustNone
End Sub

```

▶ [As it applies to the **Comments** object.](#)

This example adds a comment at the insertion point.

```

Sub AddComment()
    Selection.Collapse Direction:=wdCollapseEnd
    ActiveDocument.Comments.Add _
        Range:=Selection.Range, Text:="review this"
End Sub

```

This example adds a comment to the third paragraph in the active document.

```

Sub Comment3rd()
    Dim myRange As Range

    Set myRange = ActiveDocument.Paragraphs(3).Range
    ActiveDocument.Comments.Add Range:=myRange, _
        Text:="original third paragraph"
End Sub

```

▶ [As it applies to the **CustomLabels** object.](#)

This example adds a custom mailing label named Return Address, sets the page size, and then creates a page of these labels.

```

Sub ReturnAddrLabel()
    Dim ml As CustomLabel
    Dim addr As String

    Set ml = Application.MailingLabel.CustomLabels _
        .Add(Name:="Return Address", DotMatrix:=False)
    ml.PageSize = wdCustomLabelLetter
    addr = "Dave Edson" & vbCr & "123 Skye St." & vbCr _
        & "Our Town, WA 98004"
    Application.MailingLabel.CreateNewDocument _
        Name:="Return Address", Address:=addr, ExtractAddress:=False
End Sub

```

► [As it applies to the **Dictionaries** and **HangulHanjaConversionDictionaries** objects.](#)

This example removes all dictionaries from the list of custom spelling dictionaries and creates a new custom dictionary file. The new dictionary is assigned to be the active custom dictionary, to which new words are automatically added.

```
With CustomDictionaries
    .ClearAll
    .Add FileName:="c:\My Documents\MyCustom.dic"
    .ActiveCustomDictionary = CustomDictionaries(1)
End With
```

This example creates a new custom dictionary and assigns it to a variable. The new custom dictionary is then set to be used for text that's marked as French Canadian. Note that to run a spelling check for another language, you must have installed the proofing tools for that language.

```
Sub FrCanDic()
    Dim dicFrenchCan As Dictionary

    Set dicFrenchCan = CustomDictionaries.Add(FileName:="FrenchCanad
With dicFrenchCan
    .LanguageSpecific = True
    .LanguageID = wdFrenchCanadian
End With
End Sub
```

This example removes all dictionaries from the list of custom conversion dictionaries and creates a new custom dictionary file. The new dictionary is assigned to be the active custom dictionary, to which new words are automatically added.

```
With HangulHanjaDictionaries
    .ClearAll
    .Add FileName:="C:\My Documents\MyCustom.hhd"
    .ActiveCustomDictionary = CustomDictionaries(1)
End With
```

► [As it applies to the **Documents** object.](#)

This example creates a new document based on the Normal template.

Documents.**Add**

This example creates a new document based on the Professional Memo template.

```
Documents.Add Template:="C:\Program Files\Microsoft Office" _  
    & "\Templates\Memos\Professional Memo.dot"
```

This example creates and opens a new template, using the template attached to the active document as a model.

```
tmpName = ActiveDocument.AttachedTemplate.FullName  
Documents.Add Template:=tmpName, NewTemplate:=True
```

▶ [As it applies to the **EmailSignatureEntries** objects.](#)

This example adds an automatically numbered footnote at the end of the selection.

```
Sub NewSignature()  
    Application.EmailOptions.EmailSignature _  
        .EmailSignatureEntries.Add _  
        Name:=ActiveDocument.BuiltInDocumentProperties("Author"), _  
        Range:=Selection.Range  
End Sub
```

▶ [As it applies to the **Endnotes** and **Footnotes** objects.](#)

This example adds an automatically-numbered footnote at the end of the selection.

```
ActiveDocument.Footnotes.Add Range:= Selection.Range , _  
    Text:= "The Willow Tree, (Lone Creek Press, 1996)."
```

This example adds an endnote to the third paragraph in the active document

```
Set myRange = ActiveDocument.Paragraphs(3).Range  
ActiveDocument.Endnotes.Add Range:=myRange, _  
    Text:= "Ibid., 314."
```

This example adds a footnote that uses a custom symbol for the reference mark.

```
ActiveDocument.Footnotes.Add Range:= Selection.Range , _  
    Text:= "More information in the full report.", _  
    Reference:= "{Symbol -3998}"
```

▶ [As it applies to the **Fields** object.](#)

This example inserts a USERNAME field at the beginning of the selection.

```
Selection.Collapse Direction:=wdCollapseStart
Set myField = ActiveDocument.Fields.Add(Range:=Selection.Range, _
    Type:=wdFieldUserName)
```

This example inserts a LISTNUM field at the end of the selection. The starting switch is set to begin at 3.

```
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.Fields.Add Range:=Selection.Range, _
    Type:=wdFieldListNum, Text:="\s 3"
```

This example inserts a DATE field at the beginning of the selection and then displays the result.

```
Selection.Collapse Direction:=wdCollapseStart
Set myField = ActiveDocument.Fields.Add(Range:=Selection.Range, _
    Type:=wdFieldDate)
MsgBox myField.Result
```

▶ [As it applies to the **FirstLetterExceptions**, **OtherCorrectionsExceptions**, and **TwoInitialCapsExceptions** objects.](#)

This example adds the abbreviation addr. to the list of first-letter exceptions.

```
AutoCorrect.FirstLetterExceptions.Add Name:="addr."
```

This example adds MSOffice to the list of initial-capital exceptions.

```
AutoCorrect.TwoInitialCapsExceptions.Add Name:="MSOffice"
```

This example adds myCompany to the list of other corrections exceptions.

```
AutoCorrect.OtherCorrectionsExceptions.Add Name:="myCompany"
```

▶ [As it applies to the **FormFields** object.](#)

This example adds a check box at the end of the selection, gives it a name, and then selects it.

```

Selection.Collapse Direction:=wdCollapseEnd
Set ffield = ActiveDocument.FormFields _
    .Add(Range:=Selection.Range, Type:=wdFieldFormCheckBox)
With ffield
    .Name = "Check_Box_1"
    .CheckBox.Value = True
End With

```

▶ [As it applies to the **Frames** object.](#)

This example adds a frame around the selection.

```

ActiveDocument.Frames.Add Range:=Selection.Range

```

This example adds a frame around the third paragraph in the selection.

```

Set myFrame = Selection.Frames _
    .Add(Range:=Selection.Paragraphs(3).Range)

```

▶ [As it applies to the **HangulAndAlphabetExceptions** object.](#)

This example adds test to the list of hangul and alphabet AutoCorrect exceptions on the **Korean** tab in the **AutoCorrect Exceptions** dialog box.

```

AutoCorrect.HangulAndAlphabetExceptions.Add Name:="test"

```

▶ [As it applies to the **HeadingStyles** object.](#)

This example adds a table of contents at the beginning of the active document and then adds the Title style to the list of styles used to build a table of contents.

```

Set myToc = ActiveDocument.TablesOfContents _
    .Add(Range:=ActiveDocument.Range(0, 0), _
        UseHeadingStyles:=True, UpperHeadingLevel:=1, _
        LowerHeadingLevel:=3)
myToc.HeadingStyles.Add Style:="Title", Level:=2

```

▶ [As it applies to the **Hyperlinks** object.](#)

This example turns the selection into a hyperlink to the Microsoft address on the World Wide Web.

```

ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _
    Address:="http://www.microsoft.com"

```

This example turns the selection into a hyperlink that links to the bookmark named MyBookMark in MyFile.doc.

```
ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _  
    Address:="C:\My Documents\MyFile.doc", SubAddress:="MyBookMark"
```

This example turns the first shape in the selection into a hyperlink.

```
ActiveDocument.Hyperlinks.Add Anchor:=Selection.ShapeRange(1), _  
    Address:="http://www.microsoft.com"
```

▶ [As it applies to the **Indexes** object.](#)

This example marks an index entry, and then it creates an index at the end of the active document.

```
ActiveDocument.Indexes.MarkEntry _  
    Range:=Selection.Range, Entry:="My Entry"  
Set MyRange = ActiveDocument.Content  
MyRange.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Indexes.Add Range:=MyRange, Type:=wdIndexRunin
```

▶ [As it applies to the **KeyBindings** object.](#)

This example adds the CTRL+ALT+W key combination to the **FileClose** command. The keyboard customization is saved in the Normal template.

```
CustomizationContext = NormalTemplate  
KeyBindings.Add _  
    KeyCategory:=wdKeyCategoryCommand, _  
    Command:="FileClose", _  
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyW)
```

This example adds the ALT+F4 key combination to the Arial font and then displays the number of items in the **KeyBindings** collection. The example then clears the ALT+F4 key combination (returned it to its default setting) and redisplays the number of items in the **KeyBindings** collection.

```
CustomizationContext = ActiveDocument.AttachedTemplate  
Set myKey = KeyBindings.Add(KeyCategory:=wdKeyCategoryFont, _  
    Command:="Arial", KeyCode:=BuildKeyCode(wdKeyAlt, wdKeyF4))  
MsgBox KeyBindings.Count & " keys in KeyBindings collection"  
myKey.Clear  
MsgBox KeyBindings.Count & " keys in KeyBindings collection"
```

This example adds the CTRL+ALT+S key combination to the **Font** command with 8 points specified for the font size.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCategory:=wdKeyCategoryCommand, _
    Command:="FontSize", _
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyS), _
    CommandParameter:="8"
```

This example adds the CTRL+ALT+H key combination to the Heading 1 style in the active document.

```
CustomizationContext = ActiveDocument
KeyBindings.Add KeyCategory:=wdKeyCategoryStyle, _
    Command:="Heading 1", _
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyH)
```

This example adds the CTRL+ALT+O key combination to the AutoText entry named "Hello."

```
CustomizationContext = ActiveDocument
KeyBindings.Add KeyCategory:=wdKeyCategoryAutoText, _
    Command:="Hello", _
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyO)
```

► [As it applies to the **ListEntries** object.](#)

This example inserts a drop-down form field in the active document and then adds the items Red, Blue, and Green to the form field.

```
Set myField = ActiveDocument.FormFields.Add(Range:= _
    Selection.Range, Type:= wdFieldFormDropDown)
With myField.DropDown.ListEntries
    .Add Name:="Red"
    .Add Name:="Blue"
    .Add Name:="Green"
End With
```

► [As it applies to the **ListTemplates** object.](#)

This example adds a new, single-level list template to the active document. The example changes the numbering style for the new list template and then applies the list template to the selection.

```
Set myList = _
    ActiveDocument.ListTemplates.Add(OutlineNumbered:=False)
myList.ListLevels(1).NumberStyle = wdListNumberStyleUpperCaseLetter
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=myList
```

▶ [As it applies to the **MailMergeFields** object.](#)

This example replaces the selection with a mail merge field named MiddleInitial.

```
ActiveDocument.MailMerge.Fields.Add Range:=Selection.Range, _
    Name:="MiddleInitial"
```

▶ [As it applies to the **PageNumbers** object.](#)

This example adds a page number to the primary footer in the first section of the active document.

```
With ActiveDocument.Sections(1)
    .Footers(wdHeaderFooterPrimary).PageNumbers.Add _
        PageNumberAlignment:=wdAlignPageNumberLeft, _
        FirstPage:=True
End With
```

This example creates and formats page numbers in the header for the active document.

```
Set myPgNum = ActiveDocument.Sections(1) _
    .Headers(wdHeaderFooterPrimary) _
    .PageNumbers.Add(PageNumberAlignment:= _
        wdAlignPageNumberCenter, FirstPage:= True)
myPgNum.Select
With Selection.Range
    .Italic = True
    .Bold = True
End With
```

▶ [As it applies to the **Panes** object.](#)

The following example splits the active window such that the top pane is 30 percent of the total window size.

```
ActiveDocument.ActiveWindow.Panes.Add SplitVertical:=30
```

▶ [As it applies to the **Paragraphs** object.](#)

This example adds a paragraph after the selection.

```
Selection.Paragraphs.Add
```

This example adds a paragraph mark before the first paragraph in the selection.

```
Selection.Paragraphs.Add Range:=Selection.Paragraphs(1).Range
```

This example adds a paragraph mark before the second paragraph in the active document.

```
ActiveDocument.Paragraphs.Add _  
    Range:=ActiveDocument.Paragraphs(2).Range
```

This example adds a new paragraph mark at the end of the active document.

```
ActiveDocument.Paragraphs.Add
```

► [As it applies to the **RecentFiles** object.](#)

This example adds the active document to the list of recently used files.

```
If ActiveDocument.Saved = True Then  
    RecentFiles.Add Document:=ActiveDocument.Name  
End If
```

► [As it applies to the **Rows** object.](#)

This example inserts a new row before the first row in the selection.

```
Sub AddARow()  
    If Selection.Information(wdWithInTable) = True Then  
        Selection.Rows.Add BeforeRow:=Selection.Rows(1)  
    End If  
End Sub
```

This example adds a row to the first table and then inserts the text Cell into this row.

```
Sub CountCells()  
    Dim tblNew As Table  
    Dim rowNew As Row  
    Dim celTable As Cell  
    Dim intCount As Integer
```

```

    intCount = 1
    Set tblNew = ActiveDocument.Tables(1)
    Set rowNew = tblNew.Rows.Add(BeforeRow:=tblNew.Rows(1))
    For Each celTable In rowNew.Cells
        celTable.Range.InsertAfter Text:="Cell " & intCount
        intCount = intCount + 1
    Next celTable
End Sub

```

► [As it applies to the Sections object.](#)

This example adds a Next Page section break before the third paragraph in the active document.

```

Set myRange = ActiveDocument.Paragraphs(3).Range
ActiveDocument.Sections.Add Range:=myRange

```

This example adds a Continuous section break at the selection.

```

Set myRange = Selection.Range
ActiveDocument.Sections.Add Range:=myRange, _
    Start:=wdSectionContinuous

```

This example adds a Next Page section break at the end of the active document.

```

ActiveDocument.Sections.Add

```

► [As it applies to the Styles object.](#)

This example adds a new character style named Introduction and makes it 12-point Arial, with bold and italic formatting. The example then applies the new character style to the selection.

```

Set myStyle = ActiveDocument.Styles.Add(Name:="Introduction", _
    Type:=wdStyleTypeCharacter)
With myStyle.Font
    .Bold = True
    .Italic = True
    .Name = "Arial"
    .Size = 12
End With
Selection.Range.Style = "Introduction"

```

► [As it applies to the **Styles** object.](#)

This example adds a style sheet to the active document and places it highest in the list of style sheets attached to the document. This example assumes that you have a style sheet document named Website.css located on your C: drive.

```
Sub NewStylesheet()  
    ThisDocument.StyleSheets.Add _  
        FileName:="c:\WebSite.css", _  
        Precedence:=wdStyleSheetPrecedenceHighest  
End Sub
```

► [As it applies to the **Tables** object.](#)

This example adds a blank table with three rows and four columns at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(0, 0)  
ActiveDocument.Tables.Add Range:=myRange, NumRows:=3, NumColumns:=4
```

This example adds a new, blank table with six rows and ten columns at the end of the active document

```
Set MyRange = ActiveDocument.Content  
MyRange.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Tables.Add Range:=MyRange, NumRows:=6, _  
    NumColumns:=10
```

This example adds a table with three rows and five columns to a new document and then inserts data into each cell in the table.

```
Sub NewTable()  
    Dim docNew As Document  
    Dim tblNew As Table  
    Dim intX As Integer  
    Dim intY As Integer  
  
    Set docNew = Documents.Add  
    Set tblNew = docNew.Tables.Add(Selection.Range, 3, 5)  
    With tblNew  
        For intX = 1 To 3  
            For intY = 1 To 5  
                .Cell(intX, intY).Range.InsertAfter "Cell: R" & intX & "  
            Next intY  
        End With  
    End With
```

```
Next intX
.Columns.AutoFit
End With
End Sub
```

▶ [As it applies to the **TablesOfAuthorities** object.](#)

This example adds, at the beginning of the active document, a table of authorities that includes all categories.

```
Set myRange = ActiveDocument.Range(0, 0)
ActiveDocument.TablesOfAuthorities.Add Range:=myRange, _
    Passim:= True, Category:= 0, EntrySeparator:= ", "
```

▶ [As it applies to the **TablesOfContents** object.](#)

This example adds a table of contents at the beginning of the active document. The table of contents is built from paragraphs styled with the Heading 1, Heading 2, and Heading 3 styles or the custom styles myStyle and yourStyle.

```
Set myRange = ActiveDocument.Range(0, 0)
ActiveDocument.TablesOfContents.Add _
    Range:=myRange, _
    UseFields:=False, _
    UseHeadingStyles:=True, _
    LowerHeadingLevel:=3, _
    UpperHeadingLevel:=1, _
    AddedStyles:="myStyle, yourStyle"
```

▶ [As it applies to the **TablesOfFigures** object.](#)

This example inserts a table of figures in the active document.

```
ActiveDocument.TablesOfFigures.Add Range:=Selection.Range
```

▶ [As it applies to the **TabStops** object.](#)

This example adds a tab stop positioned at 2.5 inches (from the left edge of the page) to the selected paragraphs.

```
Selection.Paragraphs.TabStops.Add Position:=InchesToPoints(2.5)
```

This example adds two tab stops to the selected paragraphs. The first tab stop is

a left aligned, has a dotted leader, and is positioned at 1 inch (72 points) from the left edge of the page. The second tab stop is centered and is positioned at 2 inches from the left edge.

```
With Selection.Paragraphs.TabStops
    .Add Position:=InchesToPoints(1), _
        Leader:=wdTabLeaderDots, _
        Alignment:=wdAlignTabLeft
    .Add Position:=InchesToPoints(2), _
        Alignment:=wdAlignTabCenter
End With
```

▶ [As it applies to the **TextColumns** object.](#)

This example creates a new document and then adds another 2.5-inch-wide text column to it.

```
Set myDoc = Documents.Add
myDoc.PageSetup.TextColumns.Add Width:=InchesToPoints(2.5), _
    Spacing:=InchesToPoints(0.5), EvenlySpaced:=False
```

This example adds a new text column to the active document and then evenly spaces all the text columns in the document.

```
ActiveDocument.PageSetup.TextColumns.Add _
    Width:=InchesToPoints(1.5), _
    EvenlySpaced:=True
```

▶ [As it applies to the **Variables** object.](#)

This example adds a variable named Temp to the active document and then inserts a DOCVARIABLE field to display the value in the Temp variable.

```
With ActiveDocument
    .Variables.Add Name:="Temp", Value:="12"
    .Fields.Add Range:=Selection.Range, _
        Type:=wdFieldDocVariable, Text:="Temp"
End With
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
```

This example sets the value of the Blue variable to six. If this variable doesn't already exist, the example adds it to the document and sets it to six.

```
For Each aVar In ActiveDocument.Variables
```

```
    If aVar.Name = "Blue" Then num = aVar.Index
Next aVar
If num = 0 Then
    ActiveDocument.Variables.Add Name:="Blue", Value:=6
Else
    ActiveDocument.Variables(num).Value = 6
End If
```

This example stores the user name (from the **Options** dialog box) in the template attached to the active document.

```
ScreenUpdating = False
With ActiveDocument.AttachedTemplate.OpenAsDocument
    .Variables.Add Name:="UserName", Value:= Application.UserName
    .Close SaveChanges:=wdSaveChanges
End With
```

► [As it applies to the **Windows** object.](#)

This example opens a new window for the document that's displayed in the active window.

```
Windows.Add
```

This example opens a new window for MyDoc.doc.

```
Windows.Add Window:=Documents("MyDoc.doc").Windows(1)
```



AddAddress Method

Adds an entry to the address book. Each entry has values for one or more tag IDs.

expression.AddAddress(*TagID*, *Value*)

expression Required. An expression that returns an **Application** object.

TagID Required **String** array. The tag ID values for the new address entry. Each element in the array can contain one of the strings listed in the following table. Only the display name is required; the remaining entries are optional.

Tag ID	Description
PR_DISPLAY_NAME	Name displayed in the Address Book dialog box
PR_DISPLAY_NAME_PREFIX	Title (for example, "Ms." or "Dr.")
PR_GIVEN_NAME	First name
PR_SURNAME	Last name
PR_STREET_ADDRESS	Street address
PR_LOCALITY	City or locality
PR_STATE_OR_PROVINCE	State or province
PR_POSTAL_CODE	Postal code
PR_COUNTRY	Country/Region
PR_TITLE	Job title
PR_COMPANY_NAME	Company name
PR_DEPARTMENT_NAME	Department name within the company
PR_OFFICE_LOCATION	Office location
PR_PRIMARY_TELEPHONE_NUMBER	Primary telephone number
PR_PRIMARY_FAX_NUMBER	Primary fax number

PR_OFFICE_TELEPHONE_NUMBER	Office telephone number
PR_OFFICE2_TELEPHONE_NUMBER	Second office telephone number
PR_HOME_TELEPHONE_NUMBER	Home telephone number
PR_CELLULAR_TELEPHONE_NUMBER	Cellular telephone number
PR_BEEPER_TELEPHONE_NUMBER	Beeper telephone number
PR_COMMENT	Text included on the Notes tab for the address entry
PR_EMAIL_ADDRESS	Electronic mail address
PR_ADDRTYPE	Electronic mail address type
PR_OTHER_TELEPHONE_NUMBER	Alternate telephone number (other than home or office)
PR_BUSINESS_FAX_NUMBER	Business fax number
PR_HOME_FAX_NUMBER	Home fax number
PR_RADIO_TELEPHONE_NUMBER	Radio telephone number
PR_INITIALS	Initials
PR_LOCATION	Location, in the format <i>buildingnumber/roomnumber</i> (for example, 7/3007 represents room 3007 in building 7)
PR_CAR_TELEPHONE_NUMBER	Car telephone number

Value Required **String** array. The values for the new address entry. Each element corresponds to an element in the **TagID** array. For more information, see the example.

Example

This example adds an entry to the address book.

```
Dim tagIDArray(0 To 3) As String  
Dim valueArray(0 To 3) As String
```

```
tagIDArray(0) = "PR_DISPLAY_NAME"  
tagIDArray(1) = "PR_GIVEN_NAME"  
tagIDArray(2) = "PR_SURNAME"  
tagIDArray(3) = "PR_COMMENT"  
valueArray(0) = "Kim Buhler"  
valueArray(1) = "Kim"  
valueArray(2) = "Buhler"  
valueArray(3) = "This is a comment"
```

```
Application.AddAddress TagID:=tagIDArray(), Value:=valueArray()
```



AddAsk Method

-

Adds an ASK field to a mail merge main document. Returns a **MailMergeField** object. When updated, an ASK field displays a dialog box that prompts you for text to assign to the specified bookmark.

expression.**AddAsk**(*Range*, *Name*, *Prompt*, *DefaultAskText*, *AskOnce*)

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

Range Required **Range** object. The location for the ASK field.

Name Required **String**. The bookmark name that the response or default text is assigned to. Use a REF field with the bookmark name to display the result in a document.

Prompt Optional **Variant**. The text that's displayed in the dialog box.

DefaultAskText Optional **Variant**. The default response, which appears in the text box when the dialog box is displayed. Corresponds to the \d switch for an ASK field.

AskOnce Optional **Variant**. **True** to display the dialog box only once instead of each time a new data record is merged. Corresponds to the \o switch for an ASK field.

Example

This example adds an ASK field at the end of the active mail merge main document.

```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Content

rngTemp.Collapse Direction:=wdCollapseEnd
ActiveDocument.MailMerge.Fields.AddAsk _
    Range:=rngTemp, _
    Prompt:="Type your company name", _
    Name:="company", AskOnce:=True
```

This example adds an ASK field after the last mail merge field in Main.doc.

```
Dim colMailMergeFields As Object
Dim rngTemp As Range

Set colMailMergeFields = _
    Documents("Main.doc").MailMerge.Fields

colMailMergeFields(colMailMergeFields.Count).Select

Set rngTemp = Selection.Range

rngTemp.Collapse wdCollapseEnd
colMailMergeFields.AddAsk Range:=rngTemp, Name:="name", _
    Prompt:="What is your name"
```



↳ [Show All](#)

AddCallout Method

► [AddCallout method as it applies to the CanvasShapes object.](#)

Adds a borderless line callout to a drawing canvas. Returns a [Shape](#) object that represents the callout and adds it to the [CanvasShapes](#) collection.

expression.AddCallout(*Type*, *Left*, *Top*, *Width*, *Height*)

expression Required. An expression that returns a [CanvasShapes](#) object.

Type Required [MsoCalloutType](#). The type of callout.

MsoCalloutType can be one of these MsoCalloutType constants.

msoCalloutOne Positions callout line straight down along the left edge of the callout's bounding box.

msoCalloutTwo Positions callout line diagonally down and away from the left edge of the callout's bounding box.

msoCalloutThree Positions callout line straight out and then diagonally down and away from the left edge of the callout's bounding box.

msoCalloutFour Positions callout line along the left edge of the callout's bounding box.

msoCalloutMixed A return value indicating that more than one **MsoCalloutType** exists in a range or selection.

Left Required **Single**. The position, in points, of the left edge of the callout's bounding box.

Top Required **Single**. The position, in points, of the top edge of the callout's bounding box.

Width Required **Single**. The width, in points, of the callout's bounding box.

Height Required **Single**. The height, in points, of the callout's bounding box.

► [AddCallout method as it applies to the Shapes object.](#)

Adds a borderless line callout to a document. Returns a [Shape](#) object that represents the callout and adds it to the [Shapes](#) collection.

expression.**AddCallout**(*Type*, *Left*, *Top*, *Width*, *Height*, *Anchor*)

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

Type Required [MsoCalloutType](#). The type of callout.

MsoCalloutType can be one of these MsoCalloutType constants.

msoCalloutOne Positions callout line straight down along the left edge of the callout box.

msoCalloutTwo Positions callout line diagonally down and away from the left edge of the callout box.

msoCalloutThree Positions callout line straight out and then diagonally down and away from the left edge of the callout box.

msoCalloutFour Positions callout line along the left edge of the callout text box.

msoCalloutMixed A return value indicating that more than one **MsoCalloutType** exists in a range or selection.

Left Required **Single**. The position, in points, of the left edge of the callout's bounding box.

Top Required **Single**. The position, in points, of the top edge of the callout's bounding box.

Width Required **Single**. The width, in points, of the callout's bounding box.

Height Required **Single**. The height, in points, of the callout's bounding box.

Anchor Optional **Variant**. A [Range](#) object that represents the text to which the callout is bound. If *Anchor* is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the callout is positioned relative to the top and left edges of the page.

Remarks

You can insert a greater variety of callouts, such as balloons and clouds, using the [AddShape](#) method.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example adds a callout to a newly created drawing canvas.

```
Sub NewCanvasCallout()  
    Dim shpCanvas As Shape  
  
    'Add drawing canvas to the active document  
    Set shpCanvas = ActiveDocument.Shapes.AddCanvas _  
        (Left:=150, Top:=150, Width:=200, Height:=300)  
  
    'Add callout to the drawing canvas  
    shpCanvas.CanvasItems.AddCallout _  
        Type:=msoCalloutTwo, Left:=100, _  
        Top:=40, Width:=150, Height:=75  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a callout to the current document and then sets the callout angle.

```
Sub NewCallout()  
    Dim shpCallout As Shape  
  
    'Add callout to the current document  
    Set shpCallout = ThisDocument.Shapes.AddCallout( _  
        Type:=msoCalloutTwo, Left:=InchesToPoints(1.25), _  
        Top:=36, Width:=100, Height:=25)  
  
    'Add text to the callout  
    shpCallout.TextFrame.TextRange.Text = "This is a Callout."  
  
    'Format the angle of the callout line to 30 degrees  
    shpCallout.Callout.Angle = msoCalloutAngle30  
End Sub
```



↳ [Show All](#)

AddCanvas Method

Adds a [drawing canvas](#) to a document. Returns a [Shape](#) object that represents the drawing canvas and adds it to the [Shapes](#) collection.

expression.AddCanvas(Left, Top, Width, Height, Anchor)

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

Left Required **Single**. The position, in points, of the left edge of the drawing canvas, relative to the anchor.

Top Required **Single**. The position, in points, of the top edge of the drawing canvas, relative to the anchor.

Width Required **Single**. The width, in points, of the drawing canvas.

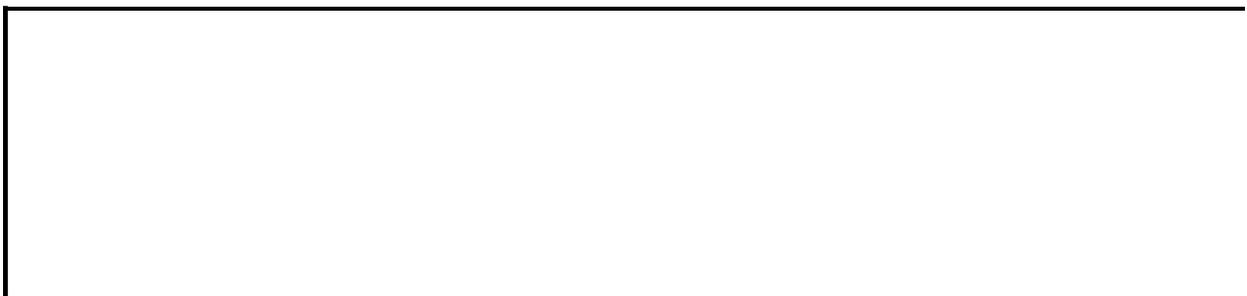
Height Required **Single**. The height, in points, of the drawing canvas.

Anchor Optional **Variant**. A [Range](#) object that represents the text to which the canvas is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the canvas is positioned relative to the top and left edges of the page.

Example

The following example adds a drawing canvas to a new document and formats the drawing canvas so it is inline with the text; then adds two shapes to the canvas, and formats the fill and line properties.

```
Sub AddInlineCanvas()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    Set docNew = Documents.Add  
  
    'Add a drawing canvas to the new document  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=150, Top:=150, Width:=70, Height:=70)  
    shpCanvas.WrapFormat.Type = wdWrapInline  
  
    'Add shapes to drawing canvas  
    With shpCanvas.CanvasItems  
        .AddShape msoShapeHeart, Left:=10, _  
            Top:=10, Width:=50, Height:=60  
        .AddLine BeginX:=0, BeginY:=0, _  
            EndX:=70, EndY:=70  
    End With  
    With shpCanvas  
        .CanvasItems(1).Fill.ForeColor _  
            .RGB = RGB(Red:=255, Green:=0, Blue:=0)  
        .CanvasItems(2).Line _  
            .EndArrowheadStyle = msoArrowheadTriangle  
    End With  
End Sub
```



↳ [Show All](#)

AddConnector Method

Returns a [Shape](#) object that represents a connecting line between two shapes in a drawing canvas.

expression.AddConnector(*Type*, *BeginX*, *BeginY*, *EndX*, *EndY*)

expression Required. An expression that returns a [CanvasShapes](#) object.

Type Required [MsoConnectorType](#). The type of connector.

MsoConnectorType can be one of these MsoConnectorType constants.

msoConnectorCurve

msoConnectorElbow

msoConnectorStraight

msoConnectorTypeMixed Not used with this method.

BeginX Required **Single**. The horizontal position that marks the beginning of the connector.

BeginY Required **Single**. The vertical position that marks the beginning of the connector.

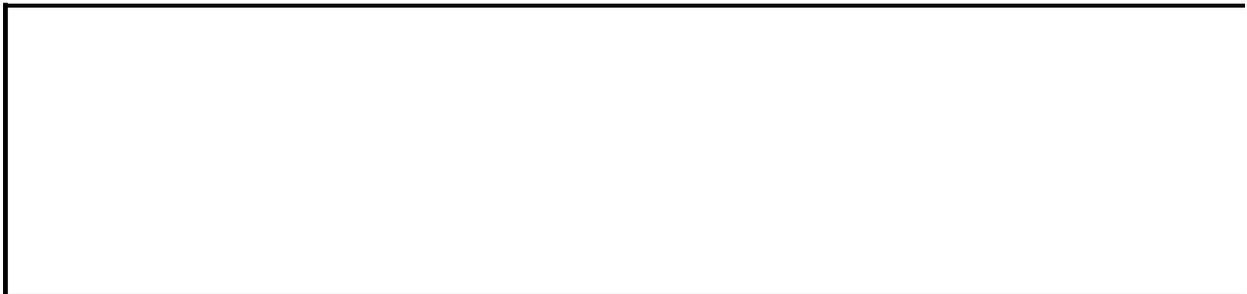
EndX Required **Single**. The horizontal position that marks the end of the connector.

EndY Required **Single**. The vertical position that marks the end of the connector.

Example

The following example adds a curved connector to a new canvas in a new document.

```
Sub AddCanvasConnector()  
  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    Set docNew = Documents.Add  
  
    'Add drawing canvas to new document  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=150, Top:=150, Width:=200, Height:=300)  
  
    'Add connector to the drawing canvas  
    shpCanvas.CanvasItems.AddConnector _  
        Type:=msoConnectorStraight, BeginX:=150, _  
        BeginY:=150, EndX:=200, EndY:=200  
  
End Sub
```



↳ [Show All](#)

AddCurve Method

▶ [AddCurve method as it applies to the CanvasShapes object.](#)

Returns a **Shape** object that represents a Bézier curve in a drawing canvas.

expression.AddCurve(**SafeArrayOfPoints**)

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

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 $3n + 1$ points, where n is the number of segments in the curve.

▶ [AddCurve method as it applies to the Shapes object.](#)

Returns a **Shape** object that represents a Bézier curve in a document.

expression.AddCurve(**SafeArrayOfPoints**, **Anchor**)

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

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 $3n + 1$ points, where n is the number of segments in the curve.

Anchor Optional **Variant**. A **Range** object that represents the text to which the curve is bound. If **Anchor** is specified, the anchor is positioned at the beginning

of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the curve is positioned relative to the top and left edges of the page.

Example

► [As it applies to the **CanvasShapes** object.](#)

This example adds a Bézier curve to a new drawing canvas.

```
Sub CanvasBezier()  
  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
    Dim sngArray(1 To 7, 1 To 2) As Single  
  
    Set docNew = Documents.Add  
  
    'Create a new drawing canvas  
    Set shpCanvas = docNew.Shapes.AddCanvas(Left:=100, _  
        Top:=100, Width:=300, Height:=50)  
  
    sngArray(1, 1) = 0  
    sngArray(1, 2) = 0  
    sngArray(2, 1) = 50  
    sngArray(2, 2) = 50  
    sngArray(3, 1) = 100  
    sngArray(3, 2) = 0  
    sngArray(4, 1) = 150  
    sngArray(4, 2) = 50  
    sngArray(5, 1) = 200  
    sngArray(5, 2) = 0  
    sngArray(6, 1) = 250  
    sngArray(6, 2) = 50  
    sngArray(7, 1) = 300  
    sngArray(7, 2) = 0  
  
    'Add Bezier curve to drawing canvas  
    shpCanvas.CanvasItems.AddCurve _  
        SafeArrayOfPoints:=sngArray  
  
End Sub
```

► [As it applies to the **Shapes** object.](#)

This example adds a two-segment Bézier curve to the active document and anchors it to the second paragraph.

```
Sub BezierCurve()  
  Dim sngArray(1 To 7, 1 To 2) As Single  
  
  sngArray(1, 1) = 0  
  sngArray(1, 2) = 0  
  sngArray(2, 1) = 72  
  sngArray(2, 2) = 72  
  sngArray(3, 1) = 100  
  sngArray(3, 2) = 40  
  sngArray(4, 1) = 20  
  sngArray(4, 2) = 50  
  sngArray(5, 1) = 90  
  sngArray(5, 2) = 120  
  sngArray(6, 1) = 60  
  sngArray(6, 2) = 30  
  sngArray(7, 1) = 150  
  sngArray(7, 2) = 90  
  
  ActiveDocument.Shapes.AddCurve _  
    SafeArrayOfPoints:=sngArray, _  
    Anchor:=ActiveDocument.Paragraphs(2).Range  
End Sub
```



↳ [Show All](#)

AddDiagram Method

Returns a [Shape](#) object that represents a newly created diagram in a document.

expression.AddDiagram(*Type*, *Left*, *Top*, *Width*, *Height*, *Anchor*)

expression Required. An expression that returns a [Shapes](#) object.

Type Required [MsoDiagramType](#).

MsoDiagramType can be one of these MsoDiagramType constants.

msoDiagramCycle Shows a process with a continuous cycle.

msoDiagramMixed Not used with this method.

msoDiagramOrgChart Shows hierarchical relationships.

msoDiagramPyramid Shows foundation-based relationships.

msoDiagramRadial Shows relationships of a core element.

msoDiagramTarget Shows steps toward a goal.

msoDiagramVenn Shows areas of overlap between elements.

Left Required **Single**. The position, measured in points, of the left edge of the diagram's bounding box relative to the anchor.

Top Required **Single**. The position, measured in points, of the top edge of the diagram's bounding box relative to the anchor.

Width Required **Single**. The width, measured in points, of the diagram's bounding box.

Height Required **Single**. The height, measured in points, of the diagram's bounding box.

Anchor Optional **Variant**. A [Range](#) object that represents the text to which the diagram is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is

omitted, the anchoring range is selected automatically and the diagram is positioned relative to the top and left edges of the page.

Example

This example adds a pyramid chart to the current document.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
    'Add first diagram node child to pyramid diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three more diagram node children to the pyramid diagram  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
End Sub
```



AddFillIn Method

Adds a FILLIN field to a mail merge main document. Returns a **MailMergeField** object. When updated, a FILLIN field displays a dialog box that prompts you for text to insert into the document at the location of the FILLIN field.

Note Use the **Add** method with the **Fields** collection object to add a FILLIN field to a document other than a mail merge main document.

expression.**AddFillIn**(*Range*, *Prompt*, *DefaultFillInText*, *AskOnce*)

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

Range Required **Range** object. The location for the FILLIN field.

Prompt Optional **Variant**. The text that's displayed in the dialog box.

DefaultFillinText Optional **Variant**. The default response, which appears in the text box when the dialog box is displayed. Corresponds to the \d switch for an FILLIN field.

AskOnce Optional **Variant**. **True** to display the prompt only once instead of each time a new data record is merged. Corresponds to the \o switch for a FILLIN field. The default value is **False**.

Example

This example adds a FILLIN field that prompts you for a name to insert after "Name:".

With Selection

```
.Collapse Direction:=wdCollapseStart
```

```
.InsertAfter "Name: "
```

```
.Collapse Direction:=wdCollapseEnd
```

End With

```
ActiveDocument.MailMerge.Fields.AddFillin Range:=Selection.Range, _  
    Prompt:="Your name?", DefaultFillInText:="Joe", AskOnce:=True
```



AddFromFile Method

Adds the specified subdocument to the master document at the start of the selection and returns a **Subdocument** object.

Note If the active view isn't either outline view or master document view, an error occurs.

expression.AddFromFile(Name, ConfirmConversions, ReadOnly, PasswordDocument, PasswordTemplate, Revert, WritePasswordDocument, WritePasswordTemplate)

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

Name Required **String**. The file name of the subdocument to be inserted into the master document.

ConfirmConversions Optional **Variant**. **True** to confirm file conversion in the **Convert File** dialog box if the file isn't in Word format.

ReadOnly Optional **Variant**. **True** to insert the subdocument as a read-only document.

PasswordDocument Optional **Variant**. The password required to open the subdocument if it's password protected.

PasswordTemplate Optional **Variant**. The password required to open the template attached to the subdocument if the template is password protected.

Revert Optional **Variant**. Controls what happens if **Name** is the file name of an open document. **True** to insert the saved version of the subdocument. **False** to insert the open version of the subdocument, which may contain unsaved changes.

WritePasswordDocument Optional **Variant**. The password required to save changes to the document file if it's write protected.

WritePasswordTemplate Optional **Variant**. The password required to save changes to the template attached to the subdocument if the template is write protected.

Example

This example adds a subdocument named "Subdoc.doc" to the active document.

```
ActiveDocument.ActiveWindow.View.Type = wdMasterView  
ActiveDocument.Subdocuments.AddFromFile _  
    Name:="C:\Subdoc.doc"
```

This example adds a password-protected subdocument named "Subdoc.doc" to the active document on a read-only basis.

```
Selection.Range.Subdocuments.AddFromFile Name:="C:\Subdoc.doc", _  
    ReadOnly:=True, PasswordDocument:="secretpassword1"
```



AddFromRange Method

-

Creates one or more subdocuments from the text in the specified range and returns a **SubDocument** object.

Note The range must begin with one of the built-in heading level styles (for example, Heading 1). Subdocuments are created at each paragraph formatted with the same heading format used at the beginning of the range. Subdocument files are saved when the master document is saved and are automatically named using text from the first line in the file.

expression.**AddFromRange**(*Range*)

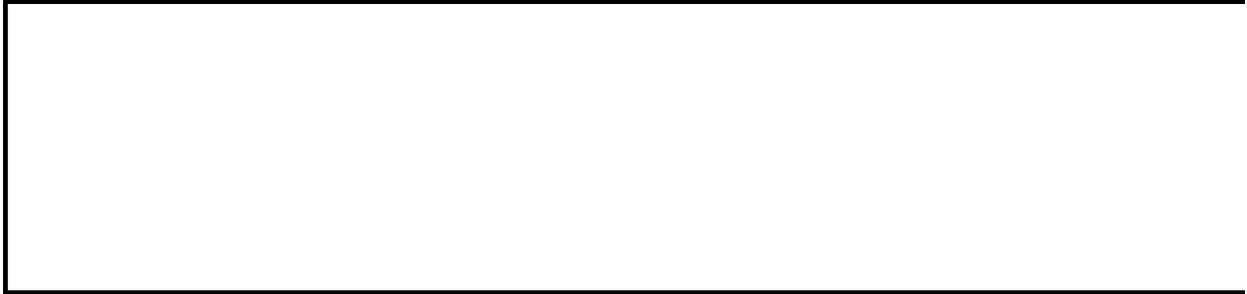
expression Required. An expression that returns a **Subdocuments** object.

Range Required **Range** object. The **Range** object used to create one or more subdocuments.

Example

This example creates one or more subdocuments from the selection.

```
ActiveDocument.ActiveWindow.View.Type = wdMasterView  
ActiveDocument.SubDocuments.AddFromRange Range:=Selection.Range
```



AddHorizontalLine Method

-
Adds a horizontal line based on an image file to the current document.

expression.**AddHorizontalLine**(*FileName*, *Range*)

expression Required. An expression that returns an **InlineShapes** object.

FileName Required **String**. The file name of the image you want to use for the horizontal line.

Range Optional **Variant**. The range above which Microsoft Word places the horizontal line. If this argument is omitted, Word places the horizontal line above the current selection.

Remarks

To add a horizontal line that isn't based on an existing image file, use the [AddHorizontalLineStandard](#) method.

Example

This example adds a horizontal line above the current selection in the active document using a file called "ArtsyRule.gif."

```
Selection.InlineShapes.AddHorizontalLine _  
    "C:\Art files\ArtsyRule.gif"
```



AddHorizontalLineStandard Method

-
Adds a horizontal line to the current document.

expression.**AddHorizontalLineStandard**(*Range*)

expression Required. An expression that returns an **InlineShapes** object.

Range Optional **Variant**. The range above which Microsoft Word places the horizontal line. If this argument is omitted, Word places the horizontal line above the current selection.

Remarks

To add a horizontal line based on an existing image file, use the [AddHorizontalLine](#) method.

Example

This example adds a horizontal line above the fifth paragraph in the active document.

```
ActiveDocument.Paragraphs(5).Range _  
    .InlineShapes.AddHorizontalLineStandard
```



AddIf Method

Adds an IF field to a mail merge main document. Returns a **MailMergeField** object. When updated, an IF field compares a field in a data record with a specified value, and then it inserts the appropriate text according to the result of the comparison.

expression.**AddIf**(*Range*, *MergeField*, *Comparison*, *CompareTo*, *TrueAutoText*, *TrueText*, *FalseAutoText*, *FalseText*)

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

Range Required **Range** object. The location for the IF field.

MergeField Required **String**. The merge field name.

Comparison Required [WdMailMergeComparison](#). The operator used in the comparison.

WdMailMergeComparison can be one of these WdMailMergeComparison constants.

wdMergeIfEqual

wdMergeIfGreaterThanOrEqual

wdMergeIfIsNotBlank

wdMergeIfLessThanOrEqual

wdMergeIfGreaterThan

wdMergeIfIsBlank

wdMergeIfLessThan

wdMergeIfNotEqual

CompareTo Optional **Variant**. The text to compare with the contents of *MergeField*.

TrueAutoText Optional **Variant**. The AutoText entry that's inserted if the

comparison is true. If this argument is specified, ***TrueText*** is ignored.

TrueText Optional **Variant**. The text that's inserted if the comparison is true.

FalseAutoText Optional **Variant**. The AutoText entry that's inserted if the comparison is false. If this argument is specified, ***FalseText*** is ignored.

FalseText Optional **Variant**. The text that's inserted if the comparison is false.

Example

This example inserts "for your personal use" if the Company merge field is blank and "for your business" if the Company merge field is not blank.

```
ActiveDocument.MailMerge.Fields.AddIf Range:=Selection.Range, _  
    MergeField:="Company", Comparison:=wdMergeIfIsBlank, _  
    TrueText:="for your personal use", _  
    FalseText:="for your business"
```



↳ [Show All](#)

AddLabel Method

► [AddLabel method as it applies to the CanvasShapes object.](#)

Adds a text label to a drawing canvas. Returns a **Shape** object that represents the drawing canvas and adds it to the **CanvasShapes** collection.

expression.AddLabel(*Orientation, Left, Top, Width, Height*)

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

Orientation Required [MsoTextOrientation](#). The orientation of the text.

MsoTextOrientation can be one of the following MsoTextOrientation constants:

msoTextOrientationDownward

msoTextOrientationHorizontal

msoTextOrientationHorizontalRotatedFarEast

msoTextOrientationMixed

msoTextOrientationUpward

msoTextOrientationVertical

msoTextOrientationVerticalFarEast

Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

Left Required **Single**. The position, measured in points, of the left edge of the label relative to the left edge of the drawing canvas.

Top Required **Single**. The position, measured in points, of the top edge of the label relative to the top edge of the drawing canvas.

Width Required **Single**. The width of the label, in points.

Height Required **Single**. The height of the label, in points.

► [AddLabel method as it applies to the Shapes object.](#)

Adds a text label to a document. Returns a **Shape** object that represents the text label and adds it to the **Shapes** collection.

expression.AddLabel(Orientation, Left, Top, Width, Height, Anchor)

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

Orientation Required [MsoTextOrientation](#). The orientation of the text.

MsoTextOrientation can be one of the following MsoTextOrientation constants:

msoTextOrientationDownward

msoTextOrientationHorizontal

msoTextOrientationHorizontalRotatedFarEast

msoTextOrientationMixed

msoTextOrientationUpward

msoTextOrientationVertical

msoTextOrientationVerticalFarEast

Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

Left Required **Single**. The position, measured in points, of the left edge of the label relative to the anchor.

Top Required **Single**. The position, measured in points, of the top edge of the label relative to the anchor.

Width Required **Single**. The width of the label, in points.

Height Required **Single**. The height of the label, in points.

Anchor Optional **Variant**. A **Range** object that represents the text to which the label is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the label is positioned relative to the top and left edges of the page.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example adds a blue text label with the text "Hello World" to a new drawing canvas in the active document.

```
Sub NewCanvasTextLabel()  
    Dim shpCanvas As Shape  
    Dim shpLabel As Shape  
  
    'Add a drawing canvas to the active document  
    Set shpCanvas = ActiveDocument.Shapes.AddCanvas _  
        (Left:=100, Top:=75, Width:=150, Height:=200)  
  
    'Add a label to the drawing canvas  
    Set shpLabel = shpCanvas.CanvasItems.AddLabel _  
        (Orientation:=msoTextOrientationHorizontal, _  
        Left:=15, Top:=15, Width:=100, Height:=100)  
  
    'Fill the label textbox with a color,  
    'add text to the label and format it  
    With shpLabel  
        With .Fill  
            .BackColor.RGB = RGB(Red:=0, Green:=0, Blue:=192)  
            'Make the fill visible  
            .Visible = msoTrue  
        End With  
        With .TextFrame.TextRange  
            .Text = "Hello World."  
            .Bold = True  
            .Font.Name = "Tahoma"  
        End With  
    End With  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a label that contains the text "Test Label" to a new document.

```
Sub NewTextLabel()  
    Dim docNew As Document  
    Dim shpLabel As Shape
```

```
Set docNew = Documents.Add

'Add label to new document
Set shpLabel = docNew.Shapes _
    .AddLabel(Orientation:=msoTextOrientationHorizontal, _
        Left:=100, Top:=100, Width:=300, Height:=200)

'Add text to the label
shpLabel.TextFrame.TextRange = "Test Label"
End Sub
```



↳ [Show All](#)

AddLine Method

▶ [AddLine method as it applies to the CanvasShapes object.](#)

Adds a line to a drawing canvas. Returns a [Shape](#) object that represents the line and adds it to the [CanvasShapes](#) collection.

expression.AddLine(**BeginX**, **BeginY**, **EndX**, **EndY**)

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

BeginX Required **Single**. The horizontal position, measured in points, of the line's starting point, relative to the drawing canvas.

BeginY Required **Single**. The vertical position, measured in points, of the line's starting point, relative to the drawing canvas.

EndX Required **Single**. The horizontal position, measured in points, of the line's end point, relative to the drawing canvas.

EndY Required **Single**. The vertical position, measured in points, of the line's end point, relative to the drawing canvas.

▶ [AddLine method as it applies to the Shapes object.](#)

Adds a line to a document. Returns a [Shape](#) object that represents the line and adds it to the [Shapes](#) collection.

expression.AddLine(**BeginX**, **BeginY**, **EndX**, **EndY**, **Anchor**)

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

BeginX Required **Single**. The horizontal position, measured in points, of the line's starting point, relative to the anchor.

BeginY Required **Single**. The vertical position, measured in points, of the line's

starting point, relative to the anchor.

EndX Required **Single**. The horizontal position, measured in points, of the line's end point, relative to the anchor.

EndY Required **Single**. The vertical position, measured in points, of the line's end point, relative to the anchor.

Anchor Optional **Variant**. A **Range** object that represents the text to which the label is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the label is positioned relative to the top and left edges of the page.

Remarks

To create an arrow, use the [Line](#) property to format a line.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example adds a purple line with an arrow to a new drawing canvas.

```
Sub NewCanvasLine()  
    Dim shpCanvas As Shape  
    Dim shpLine As Shape  
  
    'Add new drawing canvas to the active document  
    Set shpCanvas = ActiveDocument.Shapes _  
        .AddCanvas(Left:=100, Top:=75, _  
            Width:=150, Height:=200)  
  
    'Add a line to the drawing canvas  
    Set shpLine = shpCanvas.CanvasItems.AddLine( _  
        BeginX:=25, BeginY:=25, EndX:=150, EndY:=150)  
  
    'Add an arrow to the line and sets the color to purple  
    With shpLine.Line  
        .BeginArrowheadStyle = msoArrowheadDiamond  
        .BeginArrowheadWidth = msoArrowheadWide  
        .ForeColor.RGB = RGB(Red:=150, Green:=0, Blue:=255)  
    End With  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a line to the active document and then formats the line as a red arrow.

```
Sub NewLine()  
    Dim lineNew As Shape  
  
    'Add new line to document  
    Set lineNew = ActiveDocument.Shapes.AddLine_  
        (Left:=100, Top:=100, Width:=60, Height:=20)  
  
    'Format line  
    With lineNew.Line  
        .BeginArrowheadStyle = msoArrowheadNone  
        .EndArrowheadStyle = msoArrowheadTriangle  
    End With  
End Sub
```

```
        .ForeColor.RGB = RGB(Red:=128, Green:=0, Blue:=0)  
    End With  
End Sub
```



AddMergeRec Method

-

Adds a MERGEREC field to a mail merge main document. Returns a **MailMergeField** object. A MERGEREC field inserts the number of the current data record (the position of the data record in the current query result) during a mail merge.

expression.**AddMergeRec**(*Range*)

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

Range Required **Range** object. The location for the MERGEREC field.

Example

This example inserts text and a MERGEREC field at the beginning of the active document.

```
Dim rngTemp As Range
```

```
Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
```

```
ActiveDocument.MailMerge.Fields.AddMergeRec Range:=rngTemp  
rngTemp.InsertAfter "Record Number: "
```



AddMergeSeq Method

-

Adds a MERGESEQ field to a mail merge main document. Returns a **MailMergeField** object. A MERGESEQ field inserts a number based on the sequence in which data records are merged (for example, when record 50 of records 50 to 100 is merged, MERGESEQ inserts the number 1).

expression.**AddMergeSeq**(*Range*)

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

Range Required **Range** object. The location for the MERGESEQ field.

Example

This example inserts text and a MERGESEQ field at the end of the active document.

```
Dim rngTemp As Range
```

```
Set rngTemp = ActiveDocument.Content
```

```
rngTemp.Collapse Direction:=wdCollapseEnd
```

```
ActiveDocument.MailMerge.Fields.AddMergeSeq Range:=rngTemp
```

```
rngTemp.InsertAfter "Sequence Number: "
```



AddNewFrame Method

-
Adds a new frame to a frames page.

expression.**AddNewFrame**(*Where*)

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

Where Required [WdFramesetNewFrameLocation](#). Sets the location where the new frame is to be added in relation to the specified frame.

WdFramesetNewFrameLocation can be one of these
WdFramesetNewFrameLocation constants.

wdFramesetNewFrameBelow

wdFramesetNewFrameRight

wdFramesetNewFrameAbove

wdFramesetNewFrameLeft

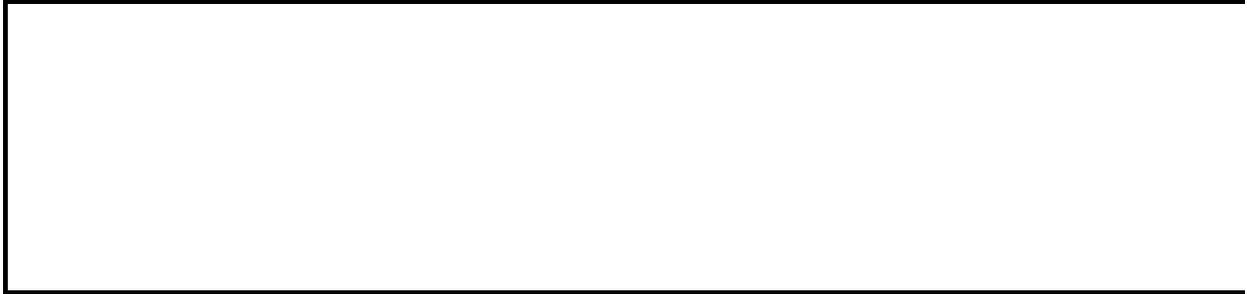
Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example adds a new frame to the immediate right of the specified frame.

```
ActiveDocument.ActiveWindow.ActivePane.FrameSet _  
    .AddNewFrame wdFramesetNewFrameRight
```



AddNext Method

-

Adds a NEXT field to a mail merge main document. Returns a **MailMergeField** object. A NEXT field advances to the next data record so that data from more than one record can be merged into the same merge document (for example, a sheet of mailing labels).

expression.**AddNext**(*Range*)

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

Range Required **Range** object. The location for the NEXT field.

Example

This example adds a NEXT field after the third MERGEFIELD field in Main.doc.

```
Documents("Main.doc").MailMerge.Fields(3).Select  
Selection.Collapse Direction:=wdCollapseEnd  
Documents("Main.doc").MailMerge.Fields.AddNext _  
    Range:=Selection.Range
```



AddNextIf Method

Adds a NEXTIF field to a mail merge main document. Returns a **MailMergeField** object. A NEXTIF field compares two expressions, and if the comparison is true, the next data record is merged into the current merge document.

expression.AddNextIf(Range, MergeField, Comparison, CompareTo)

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

Range Required **Range** object. The location for the NEXTIF field.

MergeField Required **String**. The merge field name.

Comparison Required [WdMailMergeComparison](#). The operator used in the comparison.

WdMailMergeComparison can be one of these WdMailMergeComparison constants.

wdMergeIfEqual

wdMergeIfGreaterThanOrEqual

wdMergeIfIsNotBlank

wdMergeIfLessThanOrEqual

wdMergeIfGreaterThan

wdMergeIfIsBlank

wdMergeIfLessThan

wdMergeIfNotEqual

CompareTo Required **String**. The text to compare with the contents of *MergeField*.

Example

This example adds a NEXTIF field before the first MERGEFIELD field in Main.doc. If the next postal code equals 98004, the next data record is merged into the current merge document.

```
Documents("Main.doc").MailMerge.Fields(1).Select  
Selection.Collapse Direction:=wdCollapseStart  
Documents("Main.doc").MailMerge.Fields.AddNextIf _  
    Range:=Selection.Range, MergeField:="PostalCode", _  
    Comparison:=wdMergeIfEqual, CompareTo:="98004"
```



↳ [Show All](#)

AddNode Method

▶ [AddNode method as it applies to the **DiagramNodeChildren** object.](#)

Adds a **DiagramNode** object to a collection of child diagram nodes.

expression.**AddNode**(*Index*)

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

Index Optional **Variant**. The index location of where to add the new diagram node: 0 adds before all nodes, -1 adds after all nodes, and any other **Index** number will add after that node in the collection.

▶ [AddNode method as it applies to the **DiagramNode** object.](#)

Creates a diagram node, returning a **DiagramNode** object that represents the new diagram node. For conceptual diagrams, the **DiagramNode** object is added to the end of the shapes list.

expression.**AddNode**(*Pos*)

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

Pos Optional **MsoRelativeNodePosition**. Specifies where the node will be added, relative to the calling node.

MsoRelativeNodePosition can be one of these MsoRelativeNodePosition constants.

msoAfterLastSibling

msoAfterNode *default*

msoBeforeFirstSibling

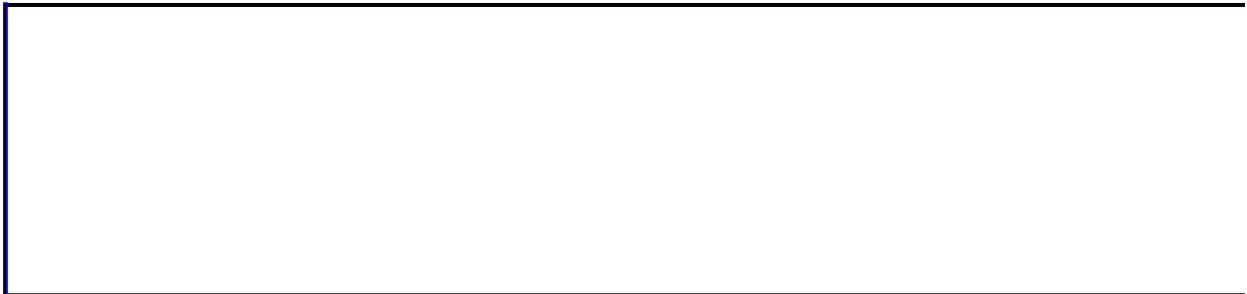
msoBeforeNode

Example

▶ [As it applies to the **DiagramNodeChildren** object.](#)

This example adds a pyramid chart to the current document and adds three nodes to the chart.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add first diagram node child to the pyramid diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three more diagram node children to the pyramid diagram  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
End Sub
```



↳ [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. To add nodes to a freeform after it's been created, use the [Insert](#) method of the [ShapeNodes](#) collection.

expression.AddNodes(*SegmentType*, *EditingType*, *X1*, *Y1*, *X2*, *Y2*, *X3*, *Y3*)

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

SegmentType Required [MsoSegmentType](#). The type of segment to be added.

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentLine

msoSegmentCurve

EditingType Required [MsoEditingType](#). The editing property of the vertex. If *SegmentType* is **msoSegmentLine**, *EditingType* must be **msoEditingAuto**.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto

msoEditingCorner

X1 Required **Single**. If the *EditingType* of the new segment is **msoEditingAuto**, this argument specifies the horizontal distance (in points) from the upper-left corner of the document to the end point of the new segment. If the *EditingType* of the new node is **msoEditingCorner**, this argument specifies the horizontal distance (in points) from the upper-left corner of the document to the first control point for the new segment.

Y1 Required **Single**. If the *EditingType* of the new segment is

msoEditingAuto, this argument specifies the vertical distance (in points) from the upper-left corner of the document to the end point of the new segment. If the *EditingType* of the new node is **msoEditingCorner**, this argument specifies the vertical distance (in points) from the upper-left corner of the document to the first control point for the new segment.

X2 Optional **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance (in points) from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the vertical distance (in points) from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance (in points) from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the vertical distance (in points) from the upper-left corner of the document 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.

Example

This example adds a freeform with five vertices to the active document.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.BuildFreeform(msoEditingCorner, 360, 200)
```

```
    .AddNodes msoSegmentCurve, msoEditingCorner, _  
        380, 230, 400, 250, 450, 300
```

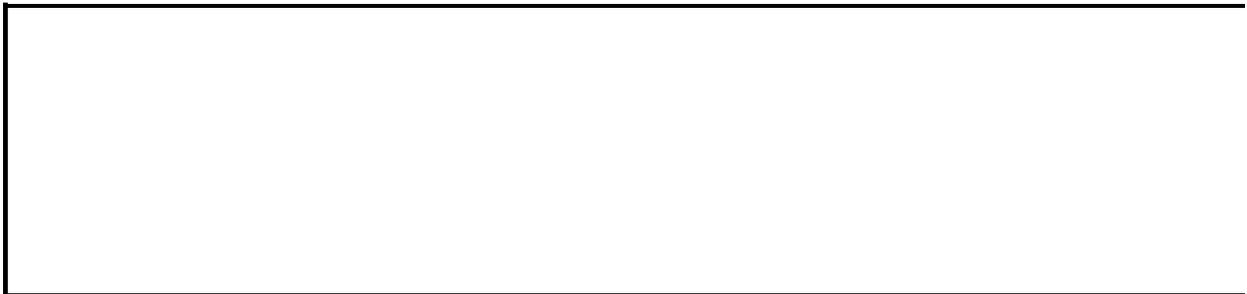
```
    .AddNodes msoSegmentCurve, msoEditingAuto, 480, 200
```

```
    .AddNodes msoSegmentLine, msoEditingAuto, 480, 400
```

```
    .AddNodes msoSegmentLine, msoEditingAuto, 360, 200
```

```
    .ConvertToShape
```

```
End With
```



↳ [Show All](#)

AddOLEControl Method

▶ [AddOLEControl method as it applies to the **InlineShapes** object.](#)

Creates an ActiveX control (formerly known as an OLE control). Returns the **InlineShape** object that represents the new ActiveX control.

expression.AddOLEControl(**ClassType**, **Range**)

expression Required. An expression that returns an **InlineShapes** object.

ClassType Optional **Variant**. The programmatic identifier for the ActiveX control to be created.

Range Optional **Variant**. The range where the ActiveX control will be placed in the text. The ActiveX control replaces the range, if the range isn't collapsed. If this argument is omitted, the Active X control is placed automatically.

▶ [AddOLEControl method as it applies to the **Shapes** object.](#)

Creates an ActiveX control (formerly known as an OLE control). Returns the **Shape** object that represents the new ActiveX control.

expression.AddOLEControl(**ClassType**, **Left**, **Top**, **Width**, **Height**, **Anchor**)

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

ClassType Optional **Variant**. The programmatic identifier for the ActiveX control to be created.

Left Optional **Variant**. The position (in points) of the left edge of the new object relative to the anchor.

Top Optional **Variant**. The position (in points) of the upper edge of the new object relative to the anchor.

Width Optional **Variant**. The width of the ActiveX control, in points.

Height Optional **Variant**. The height of the ActiveX control, in points.

Anchor Optional **Variant**. The range to which the ActiveX control is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, however, the anchor is placed automatically and the ActiveX control is positioned relative to the top and left edges of the page.

Remarks

ActiveX controls are represented as either **Shape** objects or **InlineShape** objects in Microsoft Word. To modify the properties for an ActiveX control, you use the **Object** property of the **OLEFormat** object for the specified shape or inline shape.

For information about available ActiveX control class types, see [OLE Programmatic Identifiers](#).

Example

▶ [As it applies to the **Shape** object.](#)

This example adds a check box to the active document.

```
ActiveDocument.Shapes.AddOLEControl ClassType:="Forms.CheckBox.1"
```

This example adds a combo box to the active document.

```
ActiveDocument.Shapes.AddOLEControl ClassType:="Forms.ComboBox.1"
```

This example adds a check box to the active document, clears the check box, and then adds a caption for it.

```
Set myCB = ActiveDocument.Shapes _  
    .AddOLEControl(ClassType:="Forms.CheckBox.1")  
With myCB.OLEFormat.Object  
    .Value = False  
    .Caption = "Check if over 21"  
End With
```



↳ [Show All](#)

AddOLEObject Method

► [AddOLEObject method as it applies to the **InlineShapes** object.](#)

Creates an OLE object. Returns the **InlineShape** object that represents the new OLE object.

expression.AddOLEObject(*ClassType*, *FileName*, *LinkToFile*, *DisplayAsIcon*, *IconFileName*, *IconIndex*, *IconLabel*, *Range*)

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

ClassType Optional **VARIANT**. The name of the application used to activate the specified OLE object.

FileName Optional **VARIANT**. The file from which the object is to be created. If this argument is omitted, the current folder is used. You must specify either the **ClassType** or **FileName** argument for the object, but not both.

LinkToFile Optional **VARIANT**. **True** to link the OLE object to the file from which it was created. **False** to make the OLE object an independent copy of the file. If you specified a value for **ClassType**, the **LinkToFile** argument must be **False**. The default value is **False**.

DisplayAsIcon Optional **VARIANT**. **True** to display the OLE object as an icon. The default value is **False**.

IconFileName Optional **VARIANT**. The file that contains the icon to be displayed.

IconIndex Optional **VARIANT**. The index number of the icon within **IconFileName**. The order of icons in the specified file corresponds to the order in which the icons appear in the **Change Icon** dialog box (**Insert** menu, **Object** dialog box) when the **Display as icon** check box is selected. The first icon in the file has the index number 0 (zero). If an icon with the given index number doesn't exist in **IconFileName**, the icon with the index number 1 (the second icon in the file) is used. The default value is 0 (zero).

IconLabel Optional **VARIANT**. A label (caption) to be displayed beneath the icon.

Range Optional **VARIANT**. The range where the OLE object will be placed in the text. The OLE object replaces the range, unless the range is collapsed. If this argument is omitted, the object is placed automatically.

► [AddOLEObject method as it applies to the Shapes object.](#)

Creates an OLE object. Returns the **Shape** object that represents the new OLE object.

expression.AddOLEObject(ClassType, FileName, LinkToFile, DisplayAsIcon, IconFileName, IconIndex, IconLabel, Left, Top, Width, Height, Anchor)

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

ClassType Optional **VARIANT**. The name of the application used to activate the specified OLE object.

FileName Optional **VARIANT**. The file from which the object is to be created. If this argument is omitted, the current folder is used. You must specify either the **ClassType** or **FileName** argument for the object, but not both.

LinkToFile Optional **VARIANT**. **True** to link the OLE object to the file from which it was created. **False** to make the OLE object an independent copy of the file. If you specified a value for **ClassType**, the **LinkToFile** argument must be **False**. The default value is **False**.

DisplayAsIcon Optional **VARIANT**. **True** to display the OLE object as an icon. The default value is **False**.

IconFileName Optional **VARIANT**. The file that contains the icon to be displayed.

IconIndex Optional **VARIANT**. The index number of the icon within **IconFileName**. The order of icons in the specified file corresponds to the order in which the icons appear in the **Change Icon** dialog box (**Insert** menu, **Object** dialog box) when the **Display as icon** check box is selected. The first icon in the file has the index number 0 (zero). If an icon with the given index number doesn't exist in **IconFileName**, the icon with the index number 1 (the second icon in the file) is used. The default value is 0 (zero).

IconLabel Optional **Variant**. A label (caption) to be displayed beneath the icon.

Left Optional **Variant**. The position (in points) of the left edge of the new object relative to the anchor.

Top Optional **Variant**. The position (in points) of the upper edge of the new object relative to the anchor.

Width Optional **Variant**. The width of the OLE object, in points.

Height Optional **Variant**. The height of the OLE object, in points.

Anchor Optional **Variant**. The range to which the OLE object is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph of the anchoring range. If **Anchor** is not specified, the anchor is placed automatically and the OLE Object is positioned relative to the top and left edges of the page.

Example

▶ [As it applies to the **Shapes** object.](#)

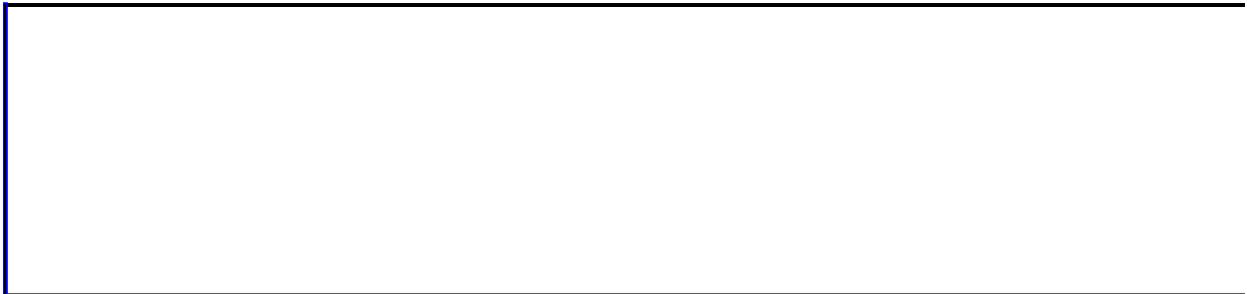
This example adds a new floating bitmap image to the active document. The bitmap is linked to another file.

```
ActiveDocument.Shapes.AddOLEObject _  
    FileName:="c:\my documents\MyDrawing.bmp", _  
    LinkToFile:=True
```

▶ [As it applies to the **InlineShapes** object.](#)

This example adds a new Microsoft Excel worksheet to the active document at the second paragraph.

```
ActiveDocument.InlineShapes.AddOLEObject _  
    ClassType:="Excel.Sheet", DisplayAsIcon:=False, _  
    Range:=ActiveDocument.Paragraphs(2).Range
```



↳ [Show All](#)

AddPicture Method

▶ [AddPicture method as it applies to the CanvasShapes object.](#)

Adds a picture to a drawing canvas. Returns a **Shape** object that represents the picture and adds it to the **CanvasShapes** collection.

expression.**AddPicture**(*FileName*, *LinkToFile*, *SaveWithDocument*, *Left*, *Top*, *Width*, *Height*)

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

FileName Required **String**. The path and file name of the picture.

LinkToFile Optional **Variant**. **True** to link the picture to the file from which it was created. **False** to make the picture an independent copy of the file. The default value is **False**.

SaveWithDocument Optional **Variant**. **True** to save the linked picture with the document. The default value is **False**.

Left Optional **Variant**. The position, measured in points, of the left edge of the new picture relative to the drawing canvas.

Top Optional **Variant**. The position, measured in points, of the top edge of the new picture relative to the drawing canvas.

Width Optional **Variant**. The width of the picture, in points.

Height Optional **Variant**. The height of the picture, in points.

▶ [AddPicture method as it applies to the InlineShapes object.](#)

Adds a picture to a document. Returns a **Shape** object that represents the picture and adds it to the **InlineShapes** collection.

expression.AddPicture(FileName, LinkToFile, SaveWithDocument, Range)

expression Required. An expression that returns an **InlineShapes** object.

FileName Required **String**. The path and file name of the picture.

LinkToFile Optional **Variant**. **True** to link the picture to the file from which it was created. **False** to make the picture an independent copy of the file. The default value is **False**.

SaveWithDocument Optional **Variant**. **True** to save the linked picture with the document. The default value is **False**.

Range Optional **Variant**. The location where the picture will be placed in the text. If the range isn't collapsed, the picture replaces the range; otherwise, the picture is inserted. If this argument is omitted, the picture is placed automatically.

▶ [AddPicture method as it applies to the Shapes object.](#)

Adds a picture to a document. Returns a **Shape** object that represents the picture and adds it to the **Shapes** collection.

expression.AddPicture(FileName, LinkToFile, SaveWithDocument, Left, Top, Width, Height, Anchor)

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

FileName Required **String**. The path and file name of the picture.

LinkToFile Optional **Variant**. **True** to link the picture to the file from which it was created. **False** to make the picture an independent copy of the file. The default value is **False**.

SaveWithDocument Optional **Variant**. **True** to save the linked picture with the document. The default value is **False**.

Left Optional **Variant**. The position, measured in points, of the left edge of the new picture relative to the anchor.

Top Optional **Variant**. The position, measured in points, of the top edge of the new picture relative to the anchor.

Width Optional **Variant**. The width of the picture, in points.

Height Optional **Variant**. The height of the picture, in points.

Anchor Optional **Variant**. The range to which the picture is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, however, the anchor is placed automatically and the picture is positioned relative to the top and left edges of the page.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

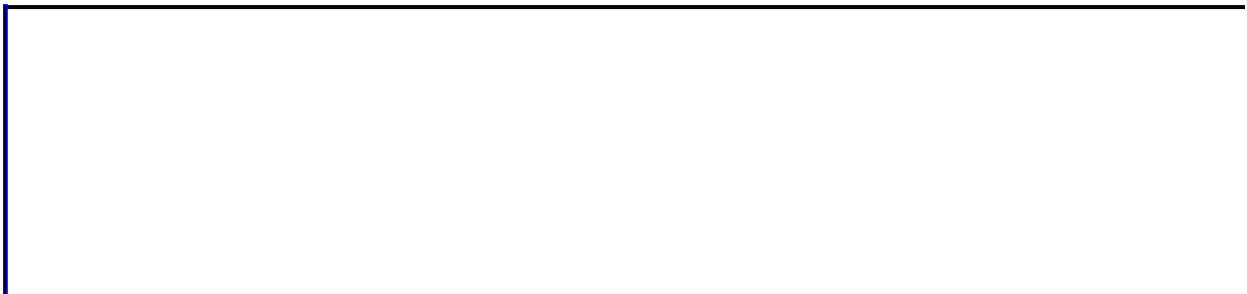
This example adds a picture to a newly created drawing canvas in the active document.

```
Sub NewCanvasPicture()  
    Dim shpCanvas As Shape  
  
    'Add a drawing canvas to the active document  
    Set shpCanvas = ActiveDocument.Shapes _  
        .AddCanvas(Left:=100, Top:=75, _  
            Width:=200, Height:=300)  
  
    'Add a graphic to the drawing canvas  
    shpCanvas.CanvasItems.AddPicture _  
        FileName:="C:\Program Files\Microsoft Office\" & _  
            "Office\Bitmaps\Styles\stone.bmp", _  
        LinkToFile:=False, SaveWithDocument:=True  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a picture to the active document. The picture is linked to the original file and is saved with the document.

```
Sub NewPicture()  
    ActiveDocument.Shapes.AddPicture _  
        FileName:="C:\Program Files\Microsoft Office\" _  
            & "Office\Bitmaps\Styles\stone.bmp", _  
        LinkToFile:=True, SaveWithDocument:=True  
End Sub
```



AddPictureBullet Method

-
Adds a picture bullet based on an image file to the current document.

expression.**AddPictureBullet**(*FileName*, *Range*)

expression Required. An expression that returns an **InlineShapes** object.

FileName Required **String**. The file name of the image you want to use for the picture bullet.

Range Optional **Variant**. The range to which Microsoft Word adds the picture bullet. Word adds the picture bullet to each paragraph in the range. If this argument is omitted, Word adds the picture bullet to each paragraph in the current selection.

Example

This example adds a picture bullet to each paragraph in the selected text using a file called "RedBullet.gif."

```
Selection.InlineShapes.AddPictureBullet _  
    "C:\Art files\RedBullet.gif"
```



↳ [Show All](#)

AddPolyline Method

▶ [AddPolyline method as it applies to the CanvasShapes object.](#)

Adds an open or closed polygon to a drawing canvas. Returns a **Shape** object that represents the polygon and adds it to the **CanvasShapes** collection.

expression.**AddPolyline**(*SafeArrayOfPoints*)

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

SafeArrayOfPoints Required **Variant**. An array of [coordinate pairs](#) that specifies the polyline drawing's vertices.

▶ [AddPolyline method as it applies to the Shapes object.](#)

Adds an open or closed polygon to a document. Returns a **Shape** object that represents the polygon and adds it to the **Shapes** collection.

expression.**AddPolyline**(*SafeArrayOfPoints*, *Anchor*)

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 drawing's vertices.

Anchor Optional **Variant**. A **Range** object that represents the text to which the polyline is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the line is positioned relative to the top and left edges of the page.

Remarks

To form a closed polygon, assign the same coordinates to the first and last vertices in the polyline drawing.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example creates a V-shaped open polyline in a new drawing canvas.

```
Sub NewCanvasPolyline()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
    Dim sngArray(1 To 3, 1 To 2) As Single  
  
    'Creates a new document and adds a drawing canvas  
    Set docNew = Documents.Add  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=100, Top:=75, Width:=200, Height:=300)  
  
    'Sets the coordinates of the array  
    sngArray(1, 1) = 100  
    sngArray(1, 2) = 75  
    sngArray(2, 1) = 150  
    sngArray(2, 2) = 100  
    sngArray(3, 1) = 100  
    sngArray(3, 2) = 125  
  
    'Adds a V-shaped open polyline to the drawing canvas  
    shpCanvas.CanvasItems.AddPolyline SafeArrayOfPoints:=sngArray  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a triangle to a new document. Because the first and last points of the triangle have the same coordinates, the polygon is closed and filled.

```
Sub NewPolyline()  
    Dim arrayTriangle(1 To 4, 1 To 2) As Single  
    Dim docNew As Document  
  
    Set docNew = Documents.Add  
  
    'Sets the coordinates of the array  
    arrayTriangle(1, 1) = 25  
    arrayTriangle(1, 2) = 100  
    arrayTriangle(2, 1) = 100
```

```
arrayTriangle(2, 2) = 150  
arrayTriangle(3, 1) = 150  
arrayTriangle(3, 2) = 50  
arrayTriangle(4, 1) = 25  
arrayTriangle(4, 2) = 100
```

'Adds a closed polygon to the document

```
docNew.Shapes.AddPolyline SafeArrayOfPoints:=arrayTriangle
```

End Sub



AddRecipient Method

-
Adds a recipient name to the specified routing slip.

Note If the recipient name isn't in the global address book, an error occurs.

expression.**AddRecipient**(*Recipient*)

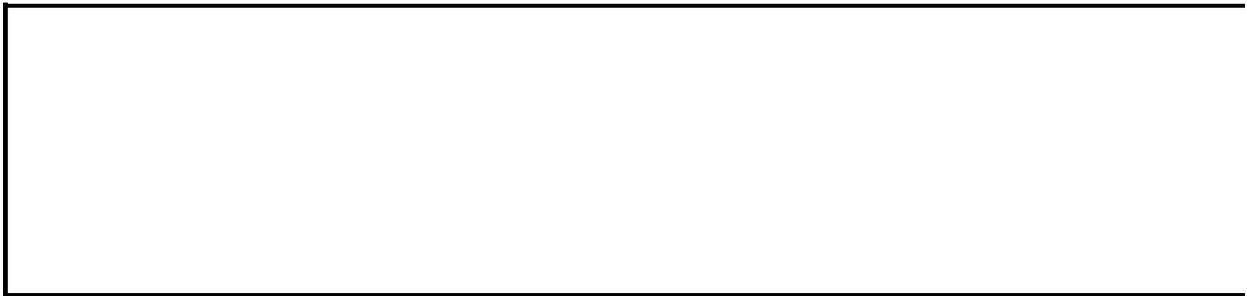
expression Required. An expression that returns a **RoutingSlip** object.

Recipient Required **String**. The recipient name.

Example

This example routes the active document to two recipients, one after the other.

```
ActiveDocument.HasRoutingSlip = True
With ActiveDocument.RoutingSlip
    .Subject = "Status Document"
    .AddRecipient Recipient:="Tim O' Brien"
    .AddRecipient Recipient:="Karin Gallagher"
    .Delivery = wdOneAfterAnother
End With
ActiveDocument.Route
```



AddRichText Method

-

Creates a formatted AutoCorrect entry, preserving all text attributes of the specified range. Returns an **AutoCorrectEntry** object. The **RichText** property for entries added by using this method returns **True**. If **AddRichText** isn't used, inserted **AutoCorrect** entries conform to the current style.

expression.**AddRichText**(*Name*, *Range*)

expression Required. An expression that returns an **AutoCorrectEntries** object.

Name Required **String**. The text to replace automatically with *Range*.

Range Required **Range** object. The formatted text that Word will insert automatically whenever *Name* is typed.

Example

This example stores the selected text as a formatted AutoCorrect entry that will be inserted automatically whenever "NewText" is typed.

```
If Selection.Type = wdSelectionNormal Then
    AutoCorrect.Entries.AddRichText "NewText", Selection.Range
Else
    MsgBox "You need to select some text."
End If
```

This example stores the third word in the active document as a formatted AutoCorrect entry that will be inserted automatically whenever "NewText" is typed.

```
AutoCorrect.Entries.AddRichText "NewText", ActiveDocument.Words(3)
```



AddSet Method

-

Adds a SET field to a mail merge main document. Returns a **MailMergeField** object. A SET field defines the text of the specified bookmark.

expression.**AddSet**(*Range*, *Name*, *ValueText*, *ValueAutoText*)

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

Range Required **Range** object. The location for the SET field.

Name Required **String**. The bookmark name that **ValueText** is assigned to.

ValueText Optional **Variant**. The text associated with the bookmark specified by the **Name** argument.

ValueAutoText Optional **Variant**. The AutoText entry that includes text associated with the bookmark specified by the **Name** argument. If this argument is specified, **ValueText** is ignored.

Example

This example adds a SET field at the beginning of the active document and then adds a REF field to display the text after the selection.

```
Dim rngTemp as Range
```

```
Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
```

```
ActiveDocument.MailMerge.Fields.AddSet Range:=rngTemp, _  
    Name:="Name", ValueText:="Joe Smith"  
Selection.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Fields.Add Range:=Selection.Range, _  
    Type:=wdFieldRef, Text:="Name"
```



↳ [Show All](#)

AddShape Method

▶ [AddShape method as it applies to the CanvasShapes object.](#)

Adds an AutoShape to a drawing canvas. Returns a **Shape** object that represents the AutoShape and adds it to the **CanvasShapes** collection.

expression.AddShape(*Type*, *Left*, *Top*, *Width*, *Height*)

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

Type Required **Long**. The type of shape to be returned. Can be any [MsoAutoShapeType](#) constant.

MsoAutoShapeType can be one of these **MsoAutoShapeType** constants.

msoShapeFlowchartDirectAccessStorage

msoShapeFlowchartDocument

msoShapeFlowchartInternalStorage

msoShapeFlowchartManualInput

msoShapeFlowchartMerge

msoShapeFlowchartOffpageConnector

msoShapeFlowchartPredefinedProcess

msoShapeFlowchartProcess

msoShapeLeftBracket

msoShapeFlowchartConnector

msoShapeFlowchartData

msoShapeFlowchartDecision

msoShapeFlowchartDelay

msoShapeFlowchartDisplay

msoShapeFlowchartExtract

msoShapeFlowchartMagneticDisk

msoShapeFlowchartManualOperation

msoShapeFlowchartMultidocument
msoShapeFlowchartOr
msoShapeFlowchartPreparation
msoShapeFlowchartPunchedTape
msoShapeFlowchartSequentialAccessStorage
msoShapeFlowchartSort
msoShapeFlowchartStoredData
msoShapeFlowchartSummingJunction
msoShapeFlowchartTerminator
msoShapeFoldedCorner
msoShapeHeart
msoShapeHexagon
msoShapeHorizontalScroll
msoShapeIsoscelesTriangle
msoShapeLeftArrow
msoShapeLeftArrowCallout
msoShapeLeftBrace
msoShapeLeftRightArrow
msoShapeLeftRightArrowCallout
msoShapeLeftRightUpArrow
msoShapeLeftUpArrow
msoShapeLightningBolt
msoShapeLineCallout1
msoShapeLineCallout1AccentBar
msoShapeLineCallout1BorderandAccentBar
msoShapeLineCallout1NoBorder
msoShapeLineCallout2
msoShapeLineCallout2AccentBar
msoShapeLineCallout2BorderandAccentBar
msoShapeLineCallout2NoBorder
msoShapeLineCallout3
msoShapeLineCallout3AccentBar
msoShapeLineCallout3BorderandAccentBar

msoShapeLineCallout3NoBorder
msoShapeLineCallout4
msoShapeLineCallout4AccentBar
msoShapeLineCallout4BorderandAccentBar
msoShapeLineCallout4NoBorder
msoShapeMixed
msoShapeMoon
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
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

Left Required **Single**. The position, measured in points, of the left edge of the AutoShape.

Top Required **Single**. The position, measured in points, of the top edge of the AutoShape.

Width Required **Single**. The width, measured in points, of the AutoShape.

Height Required **Single**. The height, measured in points, of the AutoShape.

► [AddShape method as it applies to the Shapes object.](#)

Adds an AutoShape to a document. Returns a **Shape** object that represents the AutoShape and adds it to the **Shapes** collection.

expression.AddShape(*Type, Left, Top, Width, Height, Anchor*)

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

Type Required **Long**. The type of shape to be returned. Can be any [MsoAutoShapeType](#) constant.

MsoAutoShapeType can be one of these **MsoAutoShapeType** constants.

msoShapeFlowchartDirectAccessStorage

msoShapeFlowchartDocument

msoShapeFlowchartInternalStorage

msoShapeFlowchartManualInput

msoShapeFlowchartMerge

msoShapeFlowchartOffpageConnector

msoShapeFlowchartPredefinedProcess

msoShapeFlowchartProcess

msoShapeLeftBracket

msoShapeFlowchartConnector

msoShapeFlowchartData

msoShapeFlowchartDecision

msoShapeFlowchartDelay

msoShapeFlowchartDisplay

msoShapeFlowchartExtract

msoShapeFlowchartMagneticDisk

msoShapeFlowchartManualOperation

msoShapeFlowchartMultidocument

msoShapeFlowchartOr

msoShapeFlowchartPreparation

msoShapeFlowchartPunchedTape

msoShapeFlowchartSequentialAccessStorage

msoShapeFlowchartSort

msoShapeFlowchartStoredData

msoShapeFlowchartSummingJunction
msoShapeFlowchartTerminator
msoShapeFoldedCorner
msoShapeHeart
msoShapeHexagon
msoShapeHorizontalScroll
msoShapeIsoscelesTriangle
msoShapeLeftArrow
msoShapeLeftArrowCallout
msoShapeLeftBrace
msoShapeLeftRightArrow
msoShapeLeftRightArrowCallout
msoShapeLeftRightUpArrow
msoShapeLeftUpArrow
msoShapeLightningBolt
msoShapeLineCallout1
msoShapeLineCallout1AccentBar
msoShapeLineCallout1BorderandAccentBar
msoShapeLineCallout1NoBorder
msoShapeLineCallout2
msoShapeLineCallout2AccentBar
msoShapeLineCallout2BorderandAccentBar
msoShapeLineCallout2NoBorder
msoShapeLineCallout3
msoShapeLineCallout3AccentBar
msoShapeLineCallout3BorderandAccentBar
msoShapeLineCallout3NoBorder
msoShapeLineCallout4
msoShapeLineCallout4AccentBar
msoShapeLineCallout4BorderandAccentBar
msoShapeLineCallout4NoBorder
msoShapeMixed
msoShapeMoon

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

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

Left Required **Single**. The position, measured in points, of the left edge of the AutoShape.

Top Required **Single**. The position, measured in points, of the top edge of the AutoShape.

Width Required **Single**. The width, measured in points, of the AutoShape.

Height Required **Single**. The height, measured in points, of the AutoShape.

Anchor Optional **Variant**. A **Range** object that represents the text to which the AutoShape is bound. If ***Anchor*** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the AutoShape is positioned relative to the top and left edges of the page.

Remarks

To change the type of an AutoShape that you've added, set the [AutoShapeType](#) property.

Example

▶ [As it applies to the CanvasShapes object.](#)

This example creates a new canvas in the active document and adds a circle to the canvas.

```
Sub NewCanvasShape()  
    Dim shpCanvas As Shape  
    Dim shpCanvasShape As Shape  
  
    'Add a new drawing canvas to the active document  
    Set shpCanvas = ActiveDocument.Shapes.AddCanvas( _  
        Left:=100, Top:=75, Width:=150, Height:=200)  
  
    'Add a circle to the drawing canvas  
    Set shpCanvasShape = shpCanvas.CanvasItems.AddShape( _  
        Type:=msoShapeOval, Left:=25, Top:=25, _  
        Width:=150, Height:=150)  
End Sub
```

▶ [As it applies to the Shapes object.](#)

This example adds a red rectangle to a new document.

```
Sub NewShape()  
    Dim docNew As Document  
  
    'Create a new document and adds a shape  
    Set docNew = Documents.Add  
    docNew.Shapes.AddShape Type:=msoShapeRectangle, _  
        Left:=50, Top:=50, Width:=100, Height:=200  
  
    'Format the shape  
    docNew.Shapes(1).Fill.ForeColor _  
        .RGB = RGB(Red:=200, Green:=15, Blue:=95)  
End Sub
```



AddSkipIf Method

Adds a SKIPIF field to a mail merge main document. Returns a **MailMergeField** object. A SKIPIF field compares two expressions, and if the comparison is true, SKIPIF moves to the next data record in the data source and starts a new merge document.

expression.AddSkipIf(Range, MergeField, Comparison, CompareTo)

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

Range Required **Range** object. The location for the SKIPIF field.

MergeField Required **String**. The merge field name.

Comparison Required [WdMailMergeComparison](#). The operator used in the comparison.

WdMailMergeComparison can be one of these WdMailMergeComparison constants.

wdMergeIfEqual

wdMergeIfGreaterThanOrEqual

wdMergeIfIsNotBlank

wdMergeIfLessThanOrEqual

wdMergeIfGreaterThan

wdMergeIfIsBlank

wdMergeIfLessThan

wdMergeIfNotEqual

CompareTo Optional **Variant**. The text to compare with the contents of *MergeField*.

Example

This example adds a SKIPIF field before the first MERGEFIELD field in Main.doc. If the next postal code equals 98040, the next data record is skipped.

```
Documents("Main.doc").MailMerge.Fields(1).Select  
Selection.Collapse Direction:=wdCollapseStart  
Documents("Main.doc").MailMerge.Fields.AddSkipIf _  
    Range:=Selection.Range, MergeField:="PostalCode", _  
    Comparison:=wdMergeIfEqual, CompareTo:="98040"
```



↳ [Show All](#)

AddTextbox Method

▶ [AddTextbox method as it applies to the CanvasShapes object.](#)

Adds a text box to a drawing canvas. Returns a **Shape** object that represents the text box and adds it to the **CanvasShapes** collection.

expression.AddTextbox(**Orientation**, **Left**, **Top**, **Width**, **Height**)

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

Orientation Required [MsoTextOrientation](#). The orientation of the text. Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

MsoTextOrientation can be one of these MsoTextOrientation constants.

msoTextOrientationDownward

msoTextOrientationHorizontal

msoTextOrientationHorizontalRotatedFarEast

msoTextOrientationMixed

msoTextOrientationUpward

msoTextOrientationVertical

msoTextOrientationVerticalFarEast

Left Required **Single**. The position, measured in points, of the left edge of the text box.

Top Required **Single**. The position, measured in points, of the top edge of the text box.

Width Required **Single**. The width, measured in points, of the text box.

Height Required **Single**. The height, measured in points, of the text box.

► [AddTextbox](#) method as it applies to the **Shapes** object.

Adds a text box to a document. Returns a **Shape** object that represents the text box and adds it to the **Shapes** collection.

expression.**AddTextbox**(*Orientation, Left, Top, Width, Height, Anchor*)

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

Orientation Required [MsoTextOrientation](#). The orientation of the text. Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

MsoTextOrientation can be one of these MsoTextOrientation constants.

msoTextOrientationDownward

msoTextOrientationHorizontal

msoTextOrientationHorizontalRotatedFarEast

msoTextOrientationMixed

msoTextOrientationUpward

msoTextOrientationVertical

msoTextOrientationVerticalFarEast

Left Required **Single**. The position, measured in points, of the left edge of the text box.

Top Required **Single**. The position, measured in points, of the top edge of the text box.

Width Required **Single**. The width, measured in points, of the text box.

Height Required **Single**. The height, measured in points, of the text box.

Anchor Optional **Variant**. A **Range** object that represents the text to which the text box is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the text box is positioned relative to the top and left edges of the page.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example add a textbox to a canvas in a new document.

```
Sub NewCanvasTextbox()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    'Create a new document and add a drawing canvas  
    Set docNew = Documents.Add  
    Set shpCanvas = docNew.Shapes.AddCanvas _  
        (Left:=100, Top:=75, Width:=150, Height:=200)  
  
    'Add a text box to the drawing canvas  
    shpCanvas.CanvasItems.AddTextbox _  
        Orientation:=msoTextOrientationHorizontal, _  
        Left:=1, Top:=1, Width:=100, Height:=100  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds a text box that contains the text "Test" to a new document.

```
Sub newTextbox()  
    Dim docNew As Document  
    Dim newTextbox As Shape  
  
    'Create a new document and add a text box  
    Set docNew = Documents.Add  
    Set newTextbox = docNew.Shapes.AddTextbox _  
        (Orientation:=msoTextOrientationHorizontal, _  
        Left:=100, Top:=100, Width:=300, Height:=200)  
  
    'Add text to the text box  
    newTextbox.TextFrame.TextRange = "Test"  
End Sub
```



↳ [Show All](#)

AddTextEffect Method

► [AddTextEffect method as it applies to the CanvasShapes object.](#)

Adds a WordArt shape to a drawing canvas. Returns a **Shape** object that represents the WordArt and adds it to the **CanvasShapes** collection.

expression.AddTextEffect(PresetTextEffect, Text, FontName, FontSize, FontBold, FontItalic, Left, Top)

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

PresetTextEffect Required [MsoPresetTextEffect](#). A preset text effect. 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

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

Text Required **String**. The text in the WordArt.

FontName Required **String**. The name of the font used in the WordArt.

FontSize Required **Single**. The size (in points) of the font used in the WordArt.

FontBold Required **MsoTriState**. **MsoTrue** to bold the WordArt font.

MsoTriState can be one of these **MsoTriState** constants.

msoCTrue Not used with this argument.

msoFalse Sets the font used in the WordArt to regular.

msoTriStateMixed Not used with this argument.

msoTriStateToggle Not used with this argument.

msoTrue Sets the font used in the WordArt to bold.

FontItalic Required **MsoTriState**. **MsoTrue** to italicize the WordArt font.

MsoTriState can be one of these **MsoTriState** constants.

msoCTrue Not used with this argument.

msoFalse Sets the font used in the WordArt to regular.

msoTriStateMixed Not used with this argument.

msoTriStateToggle Not used with this argument.

msoTrue Sets the font used in the WordArt to italic.

Left Required **Single**. The position, measured in points, of the left edge of the WordArt shape relative to the left edge of the drawing canvas.

Top Required **Single**. The position, measured in points, of the top edge of the WordArt shape relative to the top edge of the drawing canvas.

▶ [AddTextEffect method as it applies to the Shapes object.](#)

Adds a WordArt shape to a document. Returns a **Shape** object that represents the WordArt and adds it to the **Shapes** collection.

expression.AddTextEffect(PresetTextEffect, Text, FontName, FontSize, FontBold, FontItalic, Left, Top, Anchor)

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

PresetTextEffect Required [MsoPresetTextEffect](#). A preset text effect. 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

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

Text Required **String**. The text in the WordArt.

FontName Required **String**. The name of the font used in the WordArt.

FontSize Required **Single**. The size, in points, of the font used in the WordArt.

FontBold Required **[MsoTriState](#)**. **MsoTrue** to bold the WordArt font.

MsoTriState can be one of these **MsoTriState** constants.

msoCTrue Not used with this argument.

msoFalse Sets the font used in the WordArt to regular.

msoTriStateMixed Not used with this argument.

msoTriStateToggle Not used with this argument.

msoTrue Sets the font used in the WordArt to bold.

FontItalic Required **MsoTriState**. **MsoTrue** to italicize the WordArt font.

MsoTriState can be one of these **MsoTriState** constants.

msoCTrue Not used with this argument.

msoFalse Sets the font used in the WordArt to regular.

msoTriStateMixed Not used with this argument.

msoTriStateToggle Not used with this argument.

msoTrue Sets the font used in the WordArt to italic.

Left Required **Single**. The position, measured in points, of the left edge of the WordArt shape relative to the anchor.

Top Required **Single**. The position, measured in points, of the top edge of the WordArt shape relative to the anchor.

Anchor Optional **Variant**. A **Range** object that represents the text to which the WordArt is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the WordArt is positioned relative to the top and left edges of the page.

Remarks

When you add WordArt to a document, the height and width of the WordArt are automatically set based on the size and amount of text you specify.

Example

▶ [As it applies to the **CanvasShapes** object.](#)

This example adds a drawing canvas to a new document and inserts a WordArt shape inside the canvas that contains the text "Hello, World."

```
Sub NewCanvasTextEffect()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    'Create a new document and add a drawing canvas  
    Set docNew = Documents.Add  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=100, Top:=100, Width:=150, _  
        Height:=50)  
  
    'Add WordArt shape to the drawing canvas  
    shpCanvas.CanvasItems.AddTextEffect _  
        PresetTextEffect:=msoTextEffect20, _  
        Text:="Hello, World", FontName:="Tahoma", _  
        FontSize:=15, FontBold:=msoTrue, _  
        FontItalic:=msoFalse, _  
        Left:=120, Top:=120  
End Sub
```

▶ [As it applies to the **Shapes** object.](#)

This example adds WordArt that contains the text "This is a test" to the active document, and then it anchors the WordArt to the first paragraph.

```
Sub NewTextEffect()  
    ActiveDocument.Shapes.AddTextEffect _  
        PresetTextEffect:=msoTextEffect11, _  
        Text:="This is a test", FontName:="Arial Black", _  
        FontSize:=36, FontBold:=msoTrue, _  
        FontItalic:=msoFalse, Left:=1, Top:=1, _  
        Anchor:=ActiveDocument.Paragraphs(1).Range  
End Sub
```



AddToFavorites Method

-
Creates a shortcut to the document or hyperlink and adds it to the **Favorites** folder.

expression.**AddToFavorites**

expression Required. An expression that returns a **Document** or **Hyperlink** object.

Example

This example creates a shortcut for each hyperlink in the active document and adds it to the **Favorites** folder.

```
For Each myHyperlink In ActiveDocument.Hyperlinks
    myHyperlink.AddToFavorites
Next myHyperlink
```

This example creates a shortcut to Sales.doc and adds it to the **Favorites** folder. If Sales.doc isn't currently open, this example opens it from the C:\Documents folder.

```
For Each doc in Documents
    If LCase(doc.Name) = "sales.doc" Then isOpen = True
Next doc
If isOpen <> True Then Documents.Open _
    FileName:="C:\Documents\Sales.doc"
Documents("Sales.doc").AddToFavorites
```



After Method

Returns the next **TabStop** object to the right of *Position*.

expression.**After**(*Position*)

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

Position Required **Single**. A location on the ruler, in points.

Example

This example changes the alignment of the first custom tab stop in the first paragraph in the active document that's more than 1 inch from the left margin.

```
Dim tabTemp as TabStop
```

```
Set tabTemp = ActiveDocument.Paragraphs(1).TabStops _  
    .After(InchesToPoints(1))
```

```
tabTemp.Alignment = wdAlignTabCenter
```



Align Method

-
Aligns the shapes in the specified range of shapes.

expression.**Align**(*Align*, *RelativeTo*)

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

Align Required [MsoAlignCmd](#). Specifies the way the shapes in the specified shape range are to be aligned.

MsoAlignCmd can be one of these MsoAlignCmd constants.

msoAlignCenters

msoAlignMiddles

msoAlignTops

msoAlignBottoms

msoAlignLefts

msoAlignRights

RelativeTo Required **Long**. . **True** to align shapes relative to the edge of the document. **False** to align shapes relative to one another.

Example

This example aligns the left edges of all the shapes in the selection of shapes in myDocument with the left edge of the leftmost shape in the range.

```
Set myShapeRange = Selection.ShapeRange  
myShapeRange.Align msoAlignLefts, False
```



AppendToSpike Method

-

Deletes the specified range and adds the contents of the range to the Spike (a built-in AutoText entry). This method returns the Spike as an **AutoTextEntry** object.

expression.**AppendToSpike**(*Range*)

expression Required. An expression that returns an **AutoTextEntries** object.

Range Required **Range** object. The range that's deleted and appended to the Spike.

Remarks

The **AppendToSpike** method is only valid for the **AutoTextEntries** collection in the Normal template.

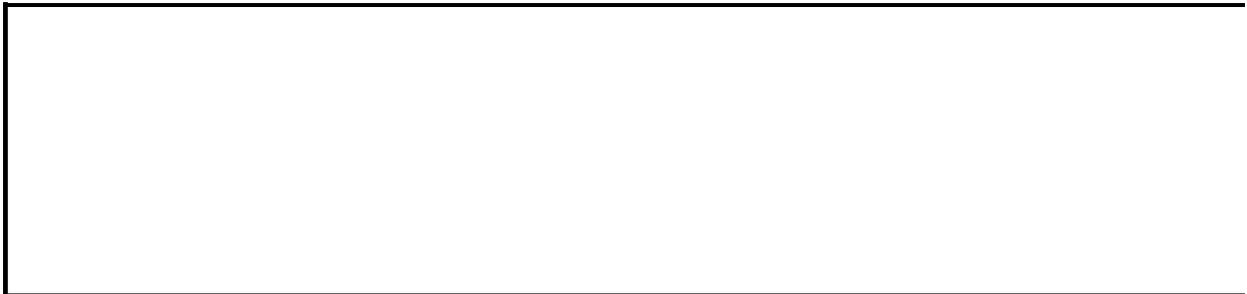
Example

This example deletes the selection and adds its contents to the Spike in the Normal template.

```
If Len(Selection.Range.Text) > 1 Then
    NormalTemplate.AutoTextEntries.AppendToSpike _
        Range:=Selection.Range
End If
```

This example clears the Spike and adds the first and third words in the active document to the Spike in the Normal template. The contents of the Spike are then inserted at the insertion point.

```
Dim atEntry As AutoTextEntry
Selection.Collapse Direction:=wdCollapseStart
For Each atEntry In NormalTemplate.AutoTextEntries
    If atEntry.Name = "Spike" Then atEntry.Delete
Next atEntry
With NormalTemplate.AutoTextEntries
    .AppendToSpike Range:=ActiveDocument.Words(3)
    .AppendToSpike Range:=ActiveDocument.Words(1)
    .Item("Spike").Insert Where:=Selection.Range
End With
```



↳ [Show All](#)

Apply Method

- ▶ [Apply method as it applies to the **AutoCorrectEntry** object.](#)

Replaces a range with the value of the specified AutoCorrect entry.

expression.**Apply**(*Range*)

expression Required. An expression that returns an **AutoCorrectEntry** object.

Range Required **Range** object.

- ▶ [Apply method as it applies to the **Shape** or **ShapeRange** object.](#)

Applies to the specified shape formatting that has been copied using the **PickUp** method.

expression.**Apply**

expression Required. An expression that returns one of the above objects.

Remarks

If formatting for the **Shape** or **ShapeRange** object has not previously been copied using the **PickUp** method, using the **Apply** method generates an error.

Example

▶ [As it applies to the **AutoCorrectEntry** object.](#)

This example adds an AutoCorrect replacement entry, then applies the "sr" AutoCorrect entry to the selected text.

```
AutoCorrect.Entries.Add Name:= "sr", Value:= "Stella Richards"  
AutoCorrect.Entries("sr").Apply Selection.Range
```

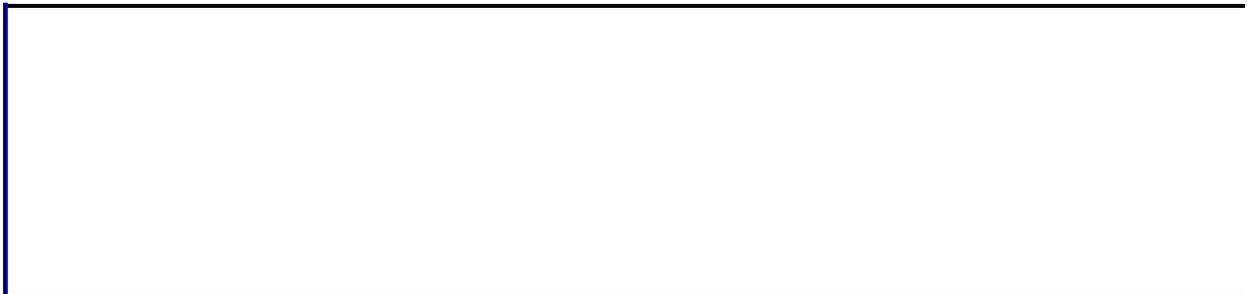
This example applies the "sr" AutoCorrect entry to the first word in the active document.

```
AutoCorrect.Entries("sr").Apply ActiveDocument.Words(1)
```

▶ [As it applies to the **Shape** object.](#)

This example copies the formatting of shape one on the active document and applies the copied formatting to shape two on the same document.

```
With ActiveDocument  
    .Shapes(1).PickUp  
    .Shapes(2).Apply  
End With
```



ApplyBulletDefault Method

-

Adds bullets and formatting to the paragraphs in the range for the specified **ListFormat** object. If the paragraphs are already formatted with bullets, this method removes the bullets and formatting.

expression.**ApplyBulletDefault**(*DefaultListBehavior*)

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

DefaultListBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word uses new Web-oriented formatting for better list display. Can be either of the following constants: **wdWord8ListBehavior** (use formatting compatible with Microsoft Word 97) or **wdWord9ListBehavior** (use Web-oriented formatting). For compatibility reasons, the default constant is **wdWord8ListBehavior**, but in new procedures you should use **wdWord9ListBehavior** to take advantage of improved Web-oriented formatting with respect to indenting and multilevel lists.

Example

This example adds bullets and formatting to the paragraphs in the selection. If there are already bullets in the selection, the example removes the bullets and formatting.

```
Selection.Range.ListFormat.ApplyBulletDefault
```

This example adds a bullet and formatting to, or removes them from, the second paragraph in MyDoc.doc.

```
Documents("MyDoc.doc").Paragraphs(2).Range.ListFormat _  
    .ApplyBulletDefault
```

This example sets the variable myRange to a range that includes paragraphs three through six of the active document, and then it checks to see whether the range contains list formatting. If there's no list formatting, default bullets are added.

```
Set myDoc = ActiveDocument  
Set myRange = myDoc.Range( _  
    Start:= myDoc.Paragraphs(3).Range.Start, _  
    End:=myDoc.Paragraphs(6).Range.End)  
If myRange.ListFormat.ListType = wdListNoNumbering Then  
    myRange.ListFormat.ApplyBulletDefault  
End If
```



↳ [Show All](#)

ApplyListTemplate Method

- ▶ [ApplyListTemplate method as it applies to the ListFormat object.](#)

Applies a set of list-formatting characteristics to the specified **ListFormat** object

expression. **ApplyListTemplate(ListTemplate, ContinuePreviousList, ApplyTo, DefaultListBehavior)**

expression Required. An expression that returns one of the above objects.

ListTemplate Required **ListTemplate** object. The list template to be applied.

ContinuePreviousList Optional **VARIANT**. **True** to continue the numbering from the previous list; **False** to start a new list.

ApplyTo Optional **VARIANT**. The portion of the list that the list template is to be applied to. Can be one of the following **WdListApplyTo** constants: **wdListApplyToSelection**, **wdListApplyToWholeList**, or **wdListApplyToThisPointForward**.

DefaultListBehavior Optional **VARIANT**. Sets a value that specifies whether Microsoft Word uses new Web-oriented formatting for better list display. Can be either of the following constants: **wdWord8ListBehavior** (use formatting compatible with Microsoft Word 97) or **wdWord9ListBehavior** (use Web-oriented formatting). For compatibility reasons, the default constant is **wdWord8ListBehavior**, but in new procedures you should use **wdWord9ListBehavior** to take advantage of improved Web-oriented formatting with respect to indenting and multilevel lists.

- ▶ [ApplyListTemplate method as it applies to the List object.](#)

Applies a set of list-formatting characteristics to the specified **List** object

expression.ApplyListTemplate(ListTemplate, ContinuePreviousList, DefaultListBehavior)

expression Required. An expression that returns one of the above objects.

ListTemplate Required **ListTemplate** object. The list template to be applied.

ContinuePreviousList Optional **Variant**. **True** to continue the numbering from the previous list; **False** to start a new list.

DefaultListBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word uses new Web-oriented formatting for better list display. Can be either of the following constants: **wdWord8ListBehavior** (use formatting compatible with Microsoft Word 97) or **wdWord9ListBehavior** (use Web-oriented formatting). For compatibility reasons, the default constant is **wdWord8ListBehavior**, but in new procedures you should use **wdWord9ListBehavior** to take advantage of improved Web-oriented formatting with respect to indenting and multilevel lists.

Example

▶ [As it applies to the **ListFormat** object.](#)

This example sets the variable `myRange` to a range in the active document, and then it checks to see whether the range has list formatting. If no list formatting has been applied, the fourth outline-numbered list template is applied to the range.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range( _
    Start:= myDoc.Paragraphs(3).Range.Start, _
    End:=myDoc.Paragraphs(6).Range.End)
If myRange.ListFormat.ListType = wdListNoNumbering Then
    myRange.ListFormat.ApplyListTemplate _
        ListTemplate:=ListGalleries(wdOutlineNumberGallery) _
        .ListTemplates(4)
End If
```

▶ [As it applies to the **List** object.](#)

This example sets the variable `myList` to the fourth list in `MyDocument.doc`, and then it applies the third bulleted list template to the list.

```
Set myList = Documents("MyDocument.doc").Lists(4)
myList.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdBulletGallery).ListTemplates(3)
```

This example sets the variable `myLstRange` to the list formatting in the second paragraph of `MyDocument.doc`. The example then applies the third numbered list template from that point forward in the list.

```
Set myLstRange = Documents("MyDocument.doc").Paragraphs(2) _
    .Range.ListFormat
myLstRange.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdNumberGallery) _
    .ListTemplates(3), _
    ApplyTo:=wdListApplyToThisPointForward
```



ApplyNumberDefault Method

-

Adds the default numbering scheme to the paragraphs in the range for the specified **ListFormat** object. If the paragraphs are already formatted as a numbered list, this method removes the numbers and formatting.

expression.**ApplyNumberDefault**(*DefaultListBehavior*)

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

DefaultListBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word uses new Web-oriented formatting for better list display. Can be either of the following constants: **wdWord8ListBehavior** (use formatting compatible with Microsoft Word 97) or **wdWord9ListBehavior** (use Web-oriented formatting). For compatibility reasons, the default constant is **wdWord8ListBehavior**, but in new procedures you should use **wdWord9ListBehavior** to take advantage of improved Web-oriented formatting with respect to indenting and multilevel lists.

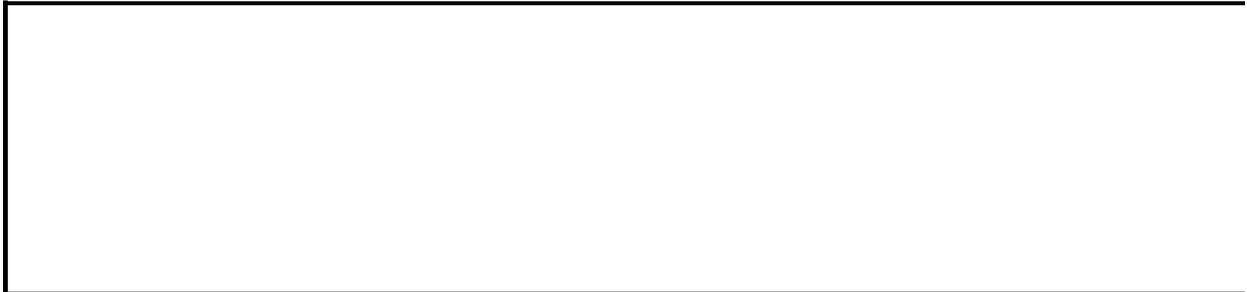
Example

This example numbers the paragraphs in the selection. If the selection is already a numbered list, the example removes the numbers and formatting.

```
Selection.Range.ListFormat.ApplyNumberDefault
```

This example sets the variable myRange to include paragraphs three through six of the active document, and then it checks to see whether the range contains list formatting. If there's no list formatting, default numbers are applied to the range.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range( _
    Start:=myDoc.Paragraphs(3).Range.Start, _
    End:=myDoc.Paragraphs(6).Range.End)
If myRange.ListFormat.ListType = wdListNoNumbering Then
    myRange.ListFormat.ApplyNumberDefault
End If
```



ApplyOutlineNumberDefault Method

-

Adds the default outline-numbering scheme to the paragraphs in the range for the specified **ListFormat** object. If the paragraphs are already formatted as an outline-numbered list, this method removes the numbers and formatting.

expression.**ApplyOutlineNumberDefault**(*DefaultListBehavior*)

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

DefaultListBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word uses new Web-oriented formatting for better list display. Can be either of the following constants: **wdWord8ListBehavior** (use formatting compatible with Microsoft Word 97) or **wdWord9ListBehavior** (use Web-oriented formatting). For compatibility reasons, the default constant is **wdWord8ListBehavior**, but in new procedures you should use **wdWord9ListBehavior** to take advantage of improved Web-oriented formatting with respect to indenting and multilevel lists.

Remarks

This method doesn't remove built-in heading styles that have been applied to paragraphs.

Example

This example adds outline numbering to the paragraphs in the selection. If the selection is already an outline-numbered list, the example removes the numbers and formatting.

```
Selection.Range.ListFormat.ApplyOutlineNumberDefault
```

This example sets the variable myRange to include paragraphs three through six of the active document, and then it checks to see whether the range contains list formatting. If there's no list formatting, the default outline-numbered list format is applied.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range( _
    Start:= myDoc.Paragraphs(3).Range.Start, _
    End:=myDoc.Paragraphs(6).Range.End)
If myRange.ListFormat.ListType = wdListNoNumbering Then
    myRange.ListFormat.ApplyOutlineNumberDefault
End If
```



ApplyPageBordersToAllSections Method

-
Applies the specified page-border formatting to all sections in a document.

expression.**ApplyPageBordersToAllSections**

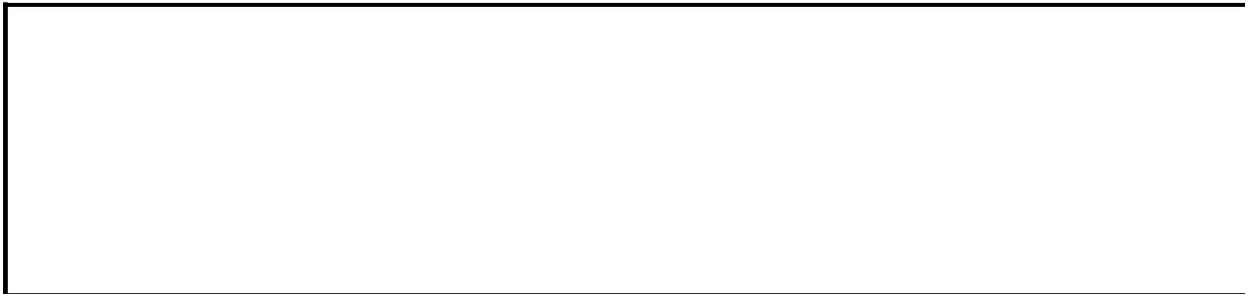
expression Required. An expression that returns a **Borders** object.

Example

This example adds a single-line page border to all sections in the active document.

```
Dim borderLoop As Border

With ActiveDocument.Sections(1)
  For Each borderLoop In .Borders
    With borderLoop
      .LineStyle = wdLineStyleSingle
      .LineWidth = wdLineWidth050pt
    End With
  Next borderLoop
  .Borders.ApplyPageBordersToAllSections
End With
```



ApplyPictureBullet Method

-
Formats a paragraph or range of paragraphs with a picture bullet.

expression.**ApplyPictureBullet**(*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 picture file.

Example

This example creates a new document with a list and applies a picture bullet format to all paragraphs except the first and last.

```
Sub ApplyPictureBulletsToParagraphs()  
    Dim docNew As Document  
    Dim rngRange As Range  
    Dim lstTemplate As ListTemplate  
    Dim intPara As Integer  
    Dim intCount As Integer  
  
    'Set the initial value of object variables  
    Set docNew = Documents.Add  
  
    'Add paragraphs to the new document, including eight list items  
    With Selection  
        .TypeText Text:="This is an introductory paragraph."  
        .TypeParagraph  
    End With  
    Do Until intPara = 8  
        With Selection  
            .TypeText Text:="This is a list item."  
            .TypeParagraph  
        End With  
        intPara = intPara + 1  
    Loop  
    Selection.TypeText Text:="This is a concluding paragraph."  
  
    'Count the total number of paragraphs in the document  
    intCount = docNew.Paragraphs.Count  
  
    'Set the range to include all paragraphs in the  
    'document except the first and the last  
    Set rngRange = docNew.Range( _  
        Start:=ActiveDocument.Paragraphs(2).Range.Start, _  
        End:=ActiveDocument.Paragraphs(intCount - 1).Range.End)  
  
    'Format the list template as a bullet  
    Set lstTemplate = ListGalleries(Index:=wdBulletGallery) _  
        .ListTemplates(7)  
  
    'Format list with a picture bullet  
    lstTemplate.ListLevels(1).ApplyPictureBullet _
```

```
FileName:="c:\pictures\bluebullet.gif"
```

```
'Apply the list format settings to the range
```

```
rngRange.ListFormat.ApplyListTemplate _
```

```
ListTemplate:=lstTemplate
```

```
End Sub
```



↳ [Show All](#)

ApplyTheme Method

Applies a [theme](#) to an open document.

expression.**ApplyTheme**(*Name*)

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

Name Required **String**. The name of the theme plus any theme formatting options you want to apply. The format of this string is "*theme nnn*" where *theme* and *nnn* are defined as follows:

String	Description
<i>theme</i>	The name of the folder that contains the data for the requested theme. (The default location for theme data folders is C:\Program Files\Common Files\Microsoft Shared\Themes.) You must use the folder name for the theme rather than the display name that appears in the Theme dialog box (Theme command, Format menu).
<i>nnn</i>	A three-digit string that indicates which theme formatting options to activate (1 to activate, 0 to deactivate). The digits correspond to the Vivid Colors , Active Graphics , and Background Image check boxes in the Theme dialog box (Theme command, Format menu). If this string is omitted, the default value for <i>nnn</i> is "011" (Active Graphics and Background Image are activated).

Example

This example applies the Artsy theme to the active document and activates the Vivid Colors option.

```
ActiveDocument.ApplyTheme "artsy 100"
```



Arrange Method

-

Arranges all open document windows in the application workspace. Because Microsoft Word uses a Single Document Interface (SDI), this method no longer has any effect.

expression.**Arrange**(*ArrangeStyle*)

expression An expression that returns a **Windows** object.

ArrangeStyle Optional **Variant**. The window arrangement. Can be either of the following **WdArrangeStyle** constants: **wdIcons** or **wdTiled**.

Example

This example arranges all open windows so that they don't overlap.

```
Windows.Arrange ArrangeStyle:=wdTiled
```

This example minimizes all open windows and then arranges the minimized windows.

```
Dim windowLoop As Window
```

```
For Each windowLoop In Windows
```

```
    With windowLoop
```

```
        .Activate
```

```
        .WindowState = wdWindowStateMinimize
```

```
    End With
```

```
Next windowLoop
```

```
Windows.Arrange ArrangeStyle:=wdIcons
```



AutoFit Method

-

Changes the width of a table column to accommodate the width of the text without changing the way text wraps in the cells.

expression.**AutoFit**

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

Remarks

If the table is already as wide as the distance between the left and right margins, this method has no affect.

Example

This example creates a 3x3 table in a new document and then changes the width of the first column to accommodate the width of the text.

```
Dim docNew as Document
Dim tableNew as Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=3)
With tableNew
    .Cell(1,1).Range.InsertAfter "First cell"
    .Columns(1).AutoFit
End With
```

This example creates a 3x3 table in a new document and then changes the width of all the columns to accommodate the width of the text.

```
Dim docNew as Document
Dim tableNew as Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 3, 3)
With tableNew
    .Cell(1,1).Range.InsertAfter "First cell"
    .Cell(1,2).Range.InsertAfter "This is cell (1,2)"
    .Cell(1,3).Range.InsertAfter "(1,3)"
    .Columns.AutoFit
End With
```



AutoFitBehavior Method

-

Determines how Microsoft Word resizes a table when the AutoFit feature is used. Word can resize the table based on the content of the table cells or the width of the document window. You can also use this method to turn off AutoFit so that the table size is fixed, regardless of cell contents or window width.

expression.**AutoFitBehavior**(*Behavior*)

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

Behavior Required [WdAutoFitBehavior](#). How Word resizes the specified table with the AutoFit feature is used.

WdAutoFitBehavior can be one of these WdAutoFitBehavior constants.

wdAutoFitContent

wdAutoFitWindow

wdAutoFitFixed

Remarks

Setting the AutoFit behavior to **wdAutoFitContent** or **wdAutoFitWindow** sets the [AllowAutoFit](#) property to **True** if it's currently **False**. Likewise, setting the AutoFit behavior to **wdAutoFitFixed** sets the **AllowAutoFit** property to **False** if it's currently **True**.

Example

This example sets the AutoFit behavior for the first table in the active document to automatically resize based on the width of the document window.

```
ActiveDocument.Tables(1).AutoFitBehavior _  
    wdAutoFitWindow
```



↳ [Show All](#)

AutoFormat Method

▶ [AutoFormat method as it applies to the Table object.](#)

Applies a predefined look to a table. The arguments for this method correspond to the options in the **Table AutoFormat** dialog box (**Table** menu).

expression.**AutoFormat**(*Format*, *ApplyBorders*, *ApplyShading*, *ApplyFont*, *ApplyColor*, *ApplyHeadingRows*, *ApplyLastRow*, *ApplyFirstColumn*, *ApplyLastColumn*, *AutoFit*)

expression Required. An expression that returns one of the above objects.

Format Optional **Variant**.

ApplyBorders Optional **Variant**. **True** to apply the border properties of the specified format. The default value is **True**.

ApplyShading Optional **Variant**. **True** to apply the shading properties of the specified format. The default value is **True**.

ApplyFont Optional **Variant**. **True** to apply the font properties of the specified format. The default value is **True**.

ApplyColor Optional **Variant**. **True** to apply the color properties of the specified format. The default value is **True**.

ApplyHeadingRows Optional **Variant**. Optional **Variant**. **True** to apply the heading-row properties of the specified format. The default value is **True**.

ApplyLastRow Optional **Variant**. **True** to apply the last-row properties of the specified format. The default value is **False**.

ApplyFirstColumn Optional **Variant**. **True** to apply the first-column properties of the specified format. The default value is **True**.

ApplyLastColumn Optional **Variant**. **True** to apply the last-column properties of the specified format. The default value is **False**.

AutoFit Optional **Variant**. **True** to decrease the width of the table columns as much as possible without changing the way text wraps in the cells. The default value is **True**.

▶ [AutoFormat method as it applies to the **Document** and **Range** objects.](#)

Automatically formats a document. Use the **Kind** property to specify a document type.

expression.**AutoFormat**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Table** object.](#)

This example creates a 5x5 table in a new document and applies all the properties of the Colorful 2 format to the table.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Range:=Selection.Range, _
    NumRows:=5, NumColumns:=5)
myTable.AutoFormat Format:=wdTableFormatColorful2
```

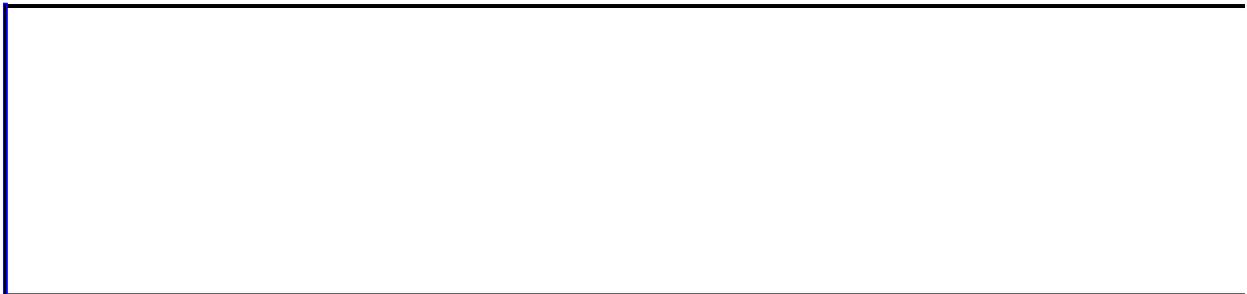
This example applies all the properties of the Classic 2 format to the table in which the insertion point is currently located. If the insertion point isn't in a table, a message box is displayed.

```
Selection.Collapse Direction:=wdCollapseStart
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).AutoFormat Format:=wdTableFormatClassic2
Else
    MsgBox "The insertion point is not in a table."
End If
```

▶ [As it applies to the **Range** object.](#)

This example automatically formats the selection.

```
Selection.Range.AutoFormat
```



AutoMarkEntries Method

-

Automatically adds XE (Index Entry) fields to the specified document, using the entries from a concordance file.

Note A concordance file is a Word document that contains a two-column table, with terms to index in the first column and index entries in the second column.

expression.**AutoMarkEntries**(*ConcordanceFileName*)

expression Required. An expression that returns an **Indexes** object.

ConcordanceFileName Required **String**. The concordance file name that includes a list of items to be indexed.

Example

This example adds index entries to Thesis.doc based on the entries in C:\Documents\List.doc.

```
Documents("Thesis.doc").Indexes.AutoMarkEntries _  
    ConcordanceFileName:="C:\Documents\List.doc"
```



AutomaticChange Method

-
Performs an **AutoFormat** action when there's a change suggested by the Office Assistant. If no AutoFormat action is active, this method generates an error.

expression.**AutomaticChange()**

expression Required. An expression that returns an **Application** object.

Example

This example completes an Office Assistant AutoFormat action if one is active.

Application.**AutomaticChange**



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. 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**).

expression.**AutomaticLength**

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

Remarks

Applying this method sets the [AutoLength](#) property to **True**.

Example

This example toggles between an automatically scaling first segment and one with a fixed length for the callout line for the first shape on the active document. For the example to work, the first shape must be a callout.

```
Dim docActive as Document

Set docActive = ActiveDocument
With docActive.Shapes(1).Callout
    If .AutoLength Then
        .CustomLength 50
    Else
        .AutomaticLength
    End If
End With
```



AutoScroll Method

-
Scrolls automatically through the specified pane.

Note This method continues to run until you stop it manually by pressing a key or clicking the mouse.

expression.**AutoScroll**(*Velocity*)

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

Velocity Required **Long**. The speed for scrolling. Can be a number from – 100 through 100. Use – 100 for full-speed backward scrolling, and use 100 for full-speed forward scrolling.

Example

This example scrolls backward through the active window pane slowly.

```
ActiveDocument.ActiveWindow.ActivePane.AutoScroll _  
    Velocity:=-20
```

This example scrolls forward through the active window pane at full speed.

```
ActiveDocument.ActiveWindow.ActivePane.AutoScroll _  
    Velocity:=100
```



AutoSum Method

-

Inserts an = (Formula) field that calculates and displays the sum of the values in table cells above or to the left of the cell specified in the expression. For information about how Word determines which values to add, see the [Formula](#) method.

expression.**AutoSum**

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

Example

This example creates a 3x3 table in a new document and sums the numbers in the first column.

```
Dim docNew as Document
Dim tableNew as Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 3, 3)

With tableNew
    .Cell(1,1).Range.InsertAfter "10"
    .Cell(2,1).Range.InsertAfter "15"
    .Cell(3,1).AutoSum
End With
```

This example sums the numbers above or to the left of the cell that contains the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells(1).AutoSum
Else
    MsgBox "The insertion point is not in a table."
End If
```



↳ [Show All](#)

AutoSummarize Method

Creates an automatic summary of the specified document, and returns a **Range** object. Corresponds to the options in **AutoSummarize** dialog box.

expression.**AutoSummarize**(*Length*, *Mode*, *UpdateProperties*)

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

Length Optional **Variant**. The length of the summary as a percentage of the total document length (the larger the number, the more detail that's included in the summary).

Mode Optional **Variant**. Specifies the way the summary is displayed. Can be one of the following **WdSummaryMode** constants.

WdSummaryMode can be one of these WdSummaryMode constants.

wdSummaryModeHighlight Highlights the key points in the specified document and displays the **AutoSummarize** toolbar.

wdSummaryModeInsert Inserts a summary at the beginning of the specified document.

wdSummaryModeCreateNew Creates a new document and inserts the summary.

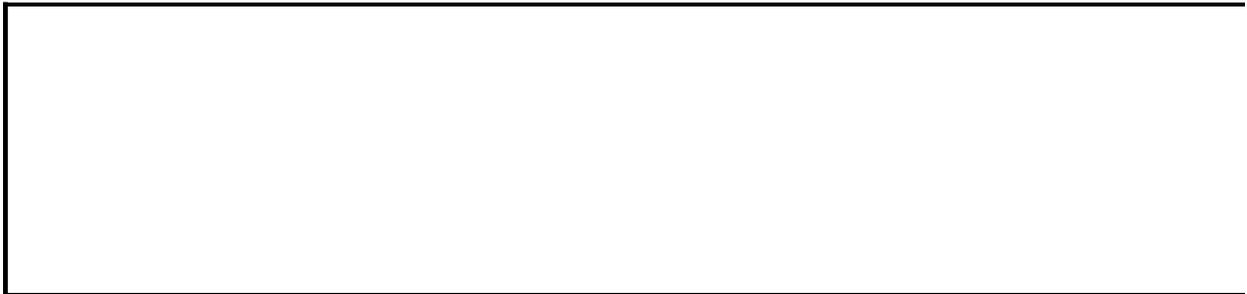
wdSummaryModeHideAllButSummary Hides everything except the summary and displays the **AutoSummarize** toolbar.

UpdateProperties Optional **Variant**. **True** to update the **Keyword** and **Comments** boxes in the **Properties** dialog box to reflect the content of the summary for the specified document.

Example

This example creates an automatic summary of the active document by highlighting its key points.

```
ActiveDocument.AutoSummarize Length:=30, _  
    Mode:=wdSummaryModeHighlight, _  
    UpdateProperties:=True
```



Before Method

Returns the next **TabStop** object to the left of *Position*.

expression.**Before**(*Position*)

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

Position Required **Single**. A location on the ruler, in points.

Example

This example changes the alignment of the first custom tab stop in the first paragraph in the active document that's less than 2 inches from the left margin.

```
Dim tsTemp As TabStop
```

```
Set tsTemp = ActiveDocument.Paragraphs(1) _  
    .TabStops.Before(InchesToPoints(2))  
tsTemp.Alignment = wdAlignTabCenter
```



↳ [Show All](#)

BoldRun Method

-

Adds the bold character format to or removes it from the current [run](#). If the run contains a mix of bold and non-bold text, this method adds the bold character format to the entire run.

expression.**BoldRun**

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

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example toggles the bold formatting for the current selection.

Selection.**BoldRun**



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 document adds a chain of three linked text boxes to it, and then breaks the link after the second text box.

```
Dim shapeTextbox1 As Shape
Dim shapeTextbox2 As Shape
Dim shapeTextbox3 As Shape

Documents.Add

Set shapeTextbox1 = ActiveDocument.Shapes.AddTextbox _
    (Orientation:=msoTextOrientationHorizontal, _
    Left:=InchesToPoints(1.5), _
    Top:=InchesToPoints(0.5), _
    Width:=InchesToPoints(1), _
    Height:=InchesToPoints(0.5))
shapeTextbox1.TextFrame.TextRange = "This is some text. " _
    & "This is some more text. This is even more text."

Set shapeTextbox2 = ActiveDocument.Shapes.AddTextbox _
    (Orientation:=msoTextOrientationHorizontal, _
    Left:=InchesToPoints(1.5), _
    Top:=InchesToPoints(1.5), _
    Width:=InchesToPoints(1), _
    Height:=InchesToPoints(0.5))

Set shapeTextbox3 = ActiveDocument.Shapes.AddTextbox _
    (Orientation:=msoTextOrientationHorizontal, _
    Left:=InchesToPoints(1.5), _
    Top:=InchesToPoints(2.5), _
    Width:=InchesToPoints(1), _
    Height:=InchesToPoints(0.5))

shapeTextbox1.TextFrame.Next = shapeTextbox2.TextFrame
shapeTextbox2.TextFrame.Next = shapeTextbox3.TextFrame
MsgBox "Textboxes 1, 2, and 3 are linked."
shapeTextbox2.TextFrame.BreakForwardLink
```



BreakLink Method

-

Breaks the link between the source file and the specified OLE object, picture, or linked field.

Note After you use this method, the link result won't be automatically updated if the source file is changed.

expression.**BreakLink**

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

Example

This example updates and then breaks the links to any shapes that are linked OLE objects in the active document.

```
Dim shapeLoop As Shape

For Each shapeLoop In ActiveDocument.Shapes
  With shapeLoop
    If .Type = msoLinkedOLEObject Then
      .LinkFormat.Update
      .LinkFormat.BreakLink
    End If
  End With
Next shapeLoop
```



BuildFreeform Method

-

Builds a freeform object. Returns a [FreeformBuilder](#) object that represents the freeform as it is being built. 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.

expression.**BuildFreeform**(*EditingType*, *X1*, *Y1*)

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

EditingType The editing property of the first node. Required [MsoEditingType](#).

MsoEditingType can be either of these MsoEditingType constants (cannot be **msoEditingSmooth** or **msoEditingSymmetric**).

msoEditingAuto

msoEditingCorner

X1, Y1 Required **Single**. The position (in points) of the first node in the freeform drawing relative to the upper-left corner of the document.

Example

This example adds a freeform with five vertices to the active document.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.BuildFreeform(msoEditingCorner, 360, 200)
```

```
    .AddNodes msoSegmentCurve, msoEditingCorner, _  
        380, 230, 400, 250, 450, 300
```

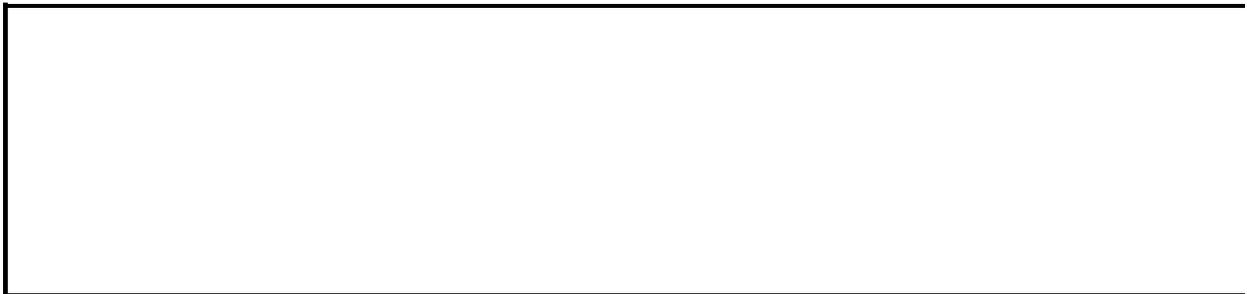
```
    .AddNodes msoSegmentCurve, msoEditingAuto, 480, 200
```

```
    .AddNodes msoSegmentLine, msoEditingAuto, 480, 400
```

```
    .AddNodes msoSegmentLine, msoEditingAuto, 360, 200
```

```
    .ConvertToShape
```

```
End With
```



BuildKeyCode Method

Returns a unique number for the specified key combination.

expression.**BuildKeyCode**(*Arg1*, *Arg2*, *Arg3*, *Arg4*)

expression Optional. An expression that returns an **Application** object.

Arg1 Required [WdKey](#). A key you specify by using one of the **WdKey** constants.

WdKey can be one of these WdKey constants.

wdKeyF

wdKeyF10

wdKeyF12

wdKeyF14

wdKeyF16

wdKeyF3

wdKeyF5

wdKeyF7

wdKeyF9

wdKeyH

wdKeyHyphen

wdKeyInsert

wdKeyK

wdKeyL

wdKeyM

wdKeyN

wdKeyNumeric0

wdKeyNumeric1

wdKeyNumeric2

wdKeyNumeric3
wdKeyNumeric4
wdKeyNumeric5
wdKeyNumeric5Special
wdKeyNumeric6
wdKeyNumeric7
wdKeyNumeric8
wdKeyNumeric9
wdKeyNumericAdd
wdKeyNumericDecimal
wdKeyNumericDivide
wdKeyNumericMultiply
wdKeyNumericSubtract
wdKeyO
wdKeyOpenSquareBrace
wdKeyOption
wdKeyP
wdKeyPageDown
wdKeyPageUp
wdKeyPause
wdKeyPeriod
wdKeyQ
wdKeyR
wdKeyReturn
wdKeyS
wdKeyScrollLock
wdKeySemiColon
wdKeyShift
wdKeySingleQuote
wdKeySlash
wdKeySpacebar
wdKeyT
wdKeyTab

wdKeyU
wdKeyV
wdKeyW
wdKeyX
wdKeyY
wdKeyZ
wdNoKey
wdKey0
wdKey1
wdKey2
wdKey3
wdKey4
wdKey5
wdKey6
wdKey7
wdKey8
wdKey9
wdKeyA
wdKeyAlt
wdKeyB
wdKeyBackSingleQuote
wdKeyBackSlash
wdKeyBackspace
wdKeyC
wdKeyCloseSquareBrace
wdKeyComma
wdKeyCommand
wdKeyControl
wdKeyD
wdKeyDelete
wdKeyE
wdKeyEnd
wdKeyEquals

wdKeyEsc
wdKeyF1
wdKeyF11
wdKeyF13
wdKeyF15
wdKeyF2
wdKeyF4
wdKeyF6
wdKeyF8
wdKeyG
wdKeyHome
wdKeyI
wdKeyJ

Arg2 - Arg4 Optional [WdKey](#). A key you specify by using one of the **WdKey** constants.

WdKey can be one of these WdKey constants.

wdKeyF
wdKeyF10
wdKeyF12
wdKeyF14
wdKeyF16
wdKeyF3
wdKeyF5
wdKeyF7
wdKeyF9
wdKeyH
wdKeyHyphen
wdKeyInsert
wdKeyK
wdKeyL
wdKeyM
wdKeyN

wdKeyNumeric0
wdKeyNumeric1
wdKeyNumeric2
wdKeyNumeric3
wdKeyNumeric4
wdKeyNumeric5
wdKeyNumeric5Special
wdKeyNumeric6
wdKeyNumeric7
wdKeyNumeric8
wdKeyNumeric9
wdKeyNumericAdd
wdKeyNumericDecimal
wdKeyNumericDivide
wdKeyNumericMultiply
wdKeyNumericSubtract
wdKeyO
wdKeyOpenSquareBrace
wdKeyOption
wdKeyP
wdKeyPageDown
wdKeyPageUp
wdKeyPause
wdKeyPeriod
wdKeyQ
wdKeyR
wdKeyReturn
wdKeyS
wdKeyScrollLock
wdKeySemiColon
wdKeyShift
wdKeySingleQuote
wdKeySlash

wdKeySpacebar
wdKeyT
wdKeyTab
wdKeyU
wdKeyV
wdKeyW
wdKeyX
wdKeyY
wdKeyZ
wdNoKey
wdKey0
wdKey1
wdKey2
wdKey3
wdKey4
wdKey5
wdKey6
wdKey7
wdKey8
wdKey9
wdKeyA
wdKeyAlt
wdKeyB
wdKeyBackSingleQuote
wdKeyBackSlash
wdKeyBackspace
wdKeyC
wdKeyCloseSquareBrace
wdKeyComma
wdKeyCommand
wdKeyControl
wdKeyD
wdKeyDelete

wdKeyE
wdKeyEnd
wdKeyEquals
wdKeyEsc
wdKeyF1
wdKeyF11
wdKeyF13
wdKeyF15
wdKeyF2
wdKeyF4
wdKeyF6
wdKeyF8
wdKeyG
wdKeyHome
wdKeyI
wdKeyJ

Example

This example assigns the ALT + F1 key combination to the **Organizer** command.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCode:=BuildKeyCode(Arg1:=wdKeyAlt, _
    Arg2:=wdKeyF1), KeyCategory:=wdKeyCategoryCommand, _
    Command:="Organizer"
```

This example removes the ALT+F1 key assignment from the Normal template.

```
CustomizationContext = NormalTemplate
FindKey(BuildKeyCode(Arg1:=wdKeyAlt, Arg2:=wdKeyF1)).Clear
```

This example displays the command assigned to the F1 key.

```
CustomizationContext = NormalTemplate
MsgBox FindKey(BuildKeyCode(Arg1:=wdKeyF1)).Command
```



Calculate Method

-
Calculates a mathematical expression within a range or selection. Returns the result as a **Single**.

expression.**Calculate**

expression Required. An expression that returns a **Range** or **Selection** object.

Example

This example inserts a mathematical expression at the beginning of the active document, calculates the expression, and then appends the results to the range. The result is "1 + 1 = 2".

```
Set myRange = ActiveDocument.Range(0, 0)
myRange.InsertBefore "1 + 1 "
myRange.InsertAfter "= " & myRange.Calculate
```

This example calculates the selected mathematical expression and displays the result.

```
MsgBox "And the answer is... " & Selection.Calculate
```



CancelAutoInsert Method

-
Prevents Word from automatically adding captions to any type of item.

expression.**CancelAutoInsert**

expression Required. An expression that returns an **AutoCaptions** object.

Example

This example prevents Word from automatically adding captions to any type of item.

`AutoCaptions.CancelAutoInsert`



CanCheckin Method

-
True if Microsoft Word can check in a specified document to a server.
Read/write **Boolean**.

expression.**CanCheckin**

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

Remarks

To take advantage of the collaboration features built into Word, documents must be stored on a Microsoft SharePoint Portal Server.

Example

This example checks the server to see if the specified document can be checked in and, if it can be, closes the document and checks it back into the server.

```
Sub CheckInOut(docCheckIn As String)
    If Documents(docCheckIn).CanCheckin = True Then
        Documents(docCheckIn).CheckIn
        MsgBox docCheckIn & " has been checked in."
    Else
        MsgBox "This file cannot be checked in " & _
            "at this time. Please try again later."
    End If
End Sub
```

To call the CheckInOut subroutine above, use the following subroutine and replace the "http://servername/workspace/report.doc" file name with an actual file located on a server mentioned in the Remarks section above.

```
Sub CheckDocInOut()
    Call CheckInOut (docCheckIn:="http://servername/workspace/report
End Sub
```



CanCheckOut Method

-
True if Microsoft Word can check out a specified document from a server.
Read/write **Boolean**.

expression.**CanCheckOut**(*FileName*)

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

FileName Required **String**. The server path and name of the document.

Remarks

To take advantage of the collaboration features built into Word, documents must be stored on a Microsoft SharePoint Portal Server.

Example

This example verifies that a document is not being edited by another user and that it can be checked out. If the document can be checked out, it copies the document to the local computer for editing.

```
Sub CheckInOut(docCheckOut As String)
    If Documents.CanCheckOut(docCheckOut) = True Then
        Documents.CheckOut docCheckOut
    Else
        MsgBox "You are unable to check out this document at this ti
    End If
End Sub
```

To call the CheckInOut subroutine, use the following subroutine and replace the "http://servername/workspace/report.doc" file name with an actual file located on a server mentioned in the Remarks section above.

```
Sub CheckDocInOut()
    Call CheckInOut (docCheckIn:="http://servername/workspace/report
End Sub
```



CanContinuePreviousList Method

Returns a **WdContinue** constant (**wdContinueDisabled**, **wdResetList**, or **wdContinueList**) that indicates whether the formatting from the previous list can be continued.

expression.**CanContinuePreviousList**(*ListTemplate*)

expression Required. An expression that returns a **List** or **ListFormat** object.

ListTemplate Required **ListTemplate** object. A list template that's been applied to previous paragraphs in the document.

Remarks

This method returns the state of the **Continue previous list** and **Restart numbering** options in the **Bullets and Numbering** dialog box for a specified list format. To change the settings of these options, set the *ContinuePreviousList* argument of the [ApplyListTemplate](#) method.

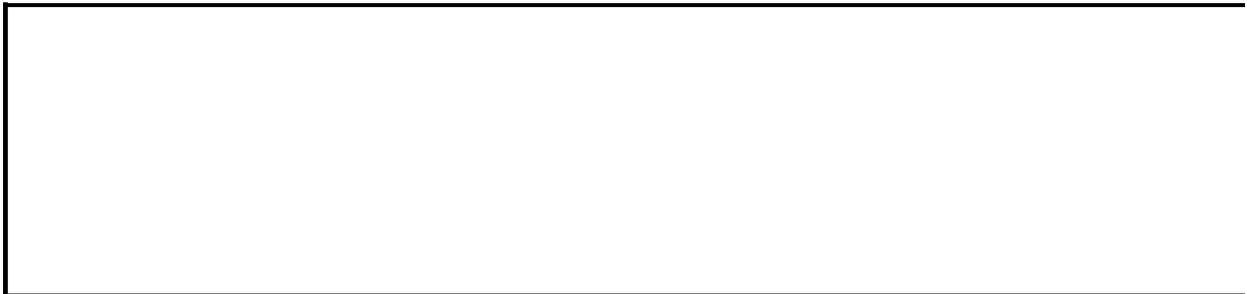
Example

This example checks to see whether numbering from a previous list is disabled. If it isn't disabled, the current list template is applied with numbering set to continue from the previous list. The selection must be within the second list, or this example creates an error.

```
Dim lfTemp As ListFormat
Dim intContinue As Integer

Set lfTemp = Selection.Range.ListFormat

intContinue = lfTemp.CanContinuePreviousList( _
    ListTemplate:=lfTemp.ListTemplate)
If intContinue <> wdContinueDisabled Then
    lfTemp.ApplyListTemplate _
        ListTemplate:=lfTemp.ListTemplate, _
        ContinuePreviousList:=True
End If
```



↳ [Show All](#)

CanvasCropBottom Method

-
Crops a percentage of the height of a [drawing canvas](#) from the bottom of the canvas.

expression.CanvasCropBottom(Increment)

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

Increment Required **Single**. The amount in percentage points of a drawing canvas's height that you want remaining after the canvas is cropped. Entering 0.9 as the increment crops ten percent of the canvas's height from the bottom. Entering 0.1 crops ninety percent of the canvas's height from the bottom.

Remark

Use the [CanvasCropTop](#) method to crop from the top.

Example

This example crops twenty-five percent of the drawing canvas's height from the bottom of the first canvas in the active document, assuming the first shape in the active document is a drawing canvas. If not, you will need to add a drawing canvas to the document using the [AddCanvas](#) method.

```
Sub CropCanvasBottom()  
    Dim shpCanvas As Shape  
  
    Set shpCanvas = ActiveDocument.Shapes(1)  
    shpCanvas.CanvasCropBottom Increment:=0.75  
End Sub
```



↳ [Show All](#)

CanvasCropLeft Method

-
Crops a percentage of the width of a [drawing canvas](#) from the left side of the canvas.

expression.CanvasCropBottom(Increment)

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

Increment Required **Single**. The amount in percentage points of the drawing canvas's width that you want remaining after the canvas is cropped. Entering 0.9 as the increment crops ten percent of the canvas's width from the left. Entering 0.1 crops ninety percent of the canvas's width from the left.

Remark

Use the [CanvasCropRight](#) method to crop from the right side of a drawing canvas.

Example

This example crops twenty-five percent of the drawing canvas's width from the left side of the first canvas in the active document, assuming the first shape in the active document is a drawing canvas. If not, you will need to add a drawing canvas to the document using the [AddCanvas](#) method.

```
Sub CropCanvasLeft()  
    Dim shpCanvas As Shape  
  
    Set shpCanvas = ActiveDocument.Shapes(1)  
    shpCanvas.CanvasCropLeft Increment:=0.75  
End Sub
```



↳ [Show All](#)

CanvasCropRight Method

-
Crops a percentage of the width of a [drawing canvas](#) from the right side of the canvas.

expression.CanvasCropBottom(*Increment*)

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

Increment Required **Single**. The amount in percentage points of the canvas's width that you want remaining after the canvas is cropped. Entering 0.9 as the increment crops ten percent of the canvas's width from the right. Entering 0.1 crops ninety percent of the canvas's width from the right.

Remark

Use the [CanvasCropLeft](#) method to crop from the left side of a drawing canvas.

Example

This example crops twenty-five percent of the drawing canvas's width from the right side of the first canvas in the active document, assuming the first shape in the active document is a drawing canvas. If not, you will need to add a drawing canvas to the document using the [AddCanvas](#) method.

```
Sub CropCanvasRight()  
    Dim shpCanvas As Shape  
  
    Set shpCanvas = ActiveDocument.Shapes(1)  
    shpCanvas.CanvasCropRight Increment:=0.75  
End Sub
```



↳ [Show All](#)

CanvasCropTop Method

-
Crops a percentage of the height of a [drawing canvas](#) from the top of the canvas.

expression.CanvasCropBottom(Increment)

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

Increment Required **Single**. The amount in percentage points of a canvas's height that you want remaining after the canvas is cropped. Entering 0.9 as the increment crops ten percent of the canvas's height from the top. Entering 0.1 crops ninety percent of the canvas's height from the top.

Remark

Use the [CanvasCropBottom](#) method to crop from the bottom.

Example

This example crops twenty-five percent of the drawing canvas's height from the top of the first canvas in the active document, assuming the first shape in the active document is a drawing canvas. If not, you will need to add a drawing canvas to the document using the [AddCanvas](#) method.

```
Sub CropCanvasTop()  
    Dim shpCanvas As Shape  
  
    Set shpCanvas = ActiveDocument.Shapes(1)  
    shpCanvas.CanvasCropTop Increment:=0.75  
End Sub
```



Cell Method

Returns a [Cell](#) object that represents a cell in a table.

expression.**Cell**(*Row*, *Column*)

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

Row Required **Long**. The number of the row in the table to return. Can be an integer between 1 and the number of rows in the table.

Column Required **Long**. The number of the cell in the table to return. Can be an integer between 1 and the number of columns in the table.

Example

This example creates a 3x3 table in a new document and inserts text into the first and last cells in the table.

```
Dim docNew As Document
Dim tableNew As Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 3, 3)

With tableNew
    .Cell(1,1).Range.InsertAfter "First cell"
    .Cell(tableNew.Rows.Count, _
        tableNew.Columns.Count).Range.InsertAfter "Last Cell"
End With
```

This example deletes the first cell from the first table in the active document.

```
If ActiveDocument.Tables.Count >= 1 Then
    ActiveDocument.Tables(1).Cell(1, 1).Delete
End If
```



CentimetersToPoints Method

-
Converts a measurement from centimeters to points (1 cm = 28.35 points).
Returns the converted measurement as a **Single**.

expression.**CentimetersToPoints**(*Centimeters*)

expression Optional. An expression that returns an **Application** object.

Centimeters Required **Single**. The centimeter value to be converted to points.

Example

This example adds a centered tab stop to all the paragraphs in the selection. The tab stop is positioned at 1.5 centimeters from the left margin.

```
Selection.Paragraphs.TabStops.Add _  
    Position:=CentimetersToPoints(1.5), _  
    Alignment:=wdAlignTabCenter
```

This example sets a first-line indent of 2.5 centimeters for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).FirstLineIndent = _  
    CentimetersToPoints(2.5)
```



ChangeFileOpenDirectory Method

Sets the folder in which Word searches for documents. The specified folder's contents are listed the next time the **Open** dialog box (**File** menu) is displayed.

Note Word searches the specified folder for documents until the user changes the folder in the **Open** dialog box or the current Word session ends. Use the [DefaultFilePath](#) property to change the default folder for documents in every Word session.

expression.**ChangeFileOpenDirectory**(*Path*)

expression Optional. An expression that returns an **Application** object.

Path Required **String**. The path to the folder in which Word searches for documents.

Example

This example changes the folder in which Word searches for documents, and then opens a file named "Test.doc."

```
ChangeFileOpenDirectory "C:\Documents"  
Documents.Open FileName:="Test.doc"
```

This example changes the folder in which Word searches for documents, and then displays the **Open** dialog box.

```
ChangeFileOpenDirectory "C:\"  
Dialogs(wdDialogFileOpen).Show
```



Check Method

-
Simulates the mail merge operation, pausing to report each error as it occurs.

expression.**Check**

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

Example

This example checks the active document for mail merge errors.

```
Dim intState As Integer
```

```
intState = ActiveDocument.MailMerge.State  
If intState = wdMainAndDataSource Or _  
    intState = wdMainAndSourceAndHeader Then  
    ActiveDocument.MailMerge.Check  
End If
```



CheckConsistency Method

-
Searches all text in a Japanese language document and displays instances where character usage is inconsistent for the same words.

expression.**CheckConsistency**

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

Example

This example checks the consistency of Japanese characters in the active document.

ActiveDocument.**CheckConsistency**



↳ [Show All](#)

CheckGrammar Method

▶ [CheckGrammar method as it applies to the **Application** object.](#)

Checks a string for grammatical errors. Returns a **Boolean** to indicate whether the string contains grammatical errors. **True** if the string contains no errors.

expression.**CheckGrammar**(*String*)

expression Required. An expression that returns an **Application** object.

String Required **String**. The string you want to check for grammatical errors.

▶ [CheckGrammar method as it applies to the **Document** and **Range** objects.](#)

Begins a spelling and grammar check for the specified document or range. If the document or range contains errors, this method displays the **Spelling and Grammar** dialog box (**Tools** menu), with the **Check grammar** check box selected. When applied to a document, this method checks all available stories (such as headers, footers, and text boxes).

expression.**CheckGrammar**

expression Required. An expression that returns a **Document** or **Range** object.

Example

▶ [As it applies to the **Document** object.](#)

This example begins a spelling and grammar check for all stories in the active document.

```
ActiveDocument.CheckGrammar
```

▶ [As it applies to the **Range** object.](#)

This example begins a spelling and grammar check on section two in MyDocument.doc.

```
Set Range2 = Documents("MyDocument.doc").Sections(2).Range  
Range2.CheckGrammar
```

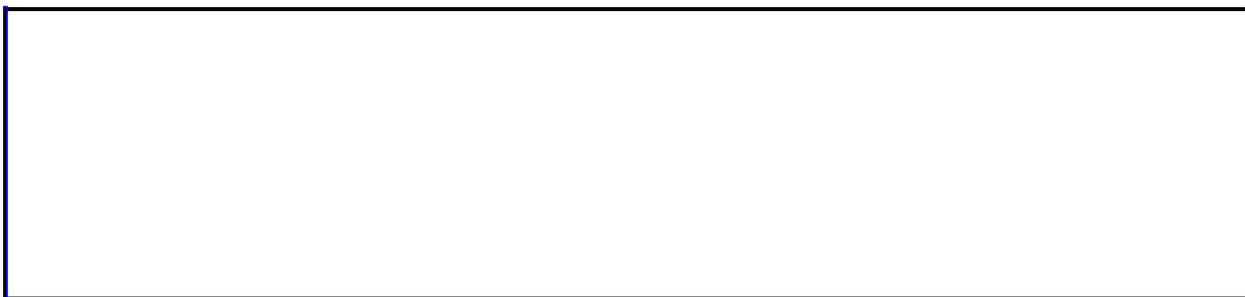
This example begins a spelling and grammar check on the selection.

```
Selection.Range.CheckGrammar
```

▶ [As it applies to the **Application** object.](#)

This example displays the result of a grammar check on the selection.

```
strPass = Application.CheckGrammar(String:=Selection.Text)  
MsgBox "Selection is grammatically correct: " & strPass
```



CheckName Method

-

Validates the e-mail addresses that appear in the **To:**, **Cc:**, and **Bcc:** lines in the active e-mail message. This method is available only if you are using Word as your e-mail editor.

Note If the names cannot be validated, the **Check Names** dialog box is displayed.

expression.**CheckName**

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

Example

This example validates the e-mail addresses that appear in the active e-mail message.

`Application.MailMessage.CheckName`



CheckNewSmartTags Method

-
Accesses the Microsoft Office Web site for available smart tag recognizer and action files.

expression.**CheckNewSmartTags**

expression Required. An expression that returns a [Document](#) object.

Remarks

The **CheckNewSmartTags** method is equivalent to clicking the **More Smart Tags** button on the **Smart Tags** tab of the **AutoCorrect** dialog box (**Tools** menu).

Example

This example displays the Office Web site for smart tags.

```
Sub GetNewSmartTagFiles()  
    ThisDocument.CheckNewSmartTags  
End Sub
```



CheckOut Method

-
Copies a specified document from a server to a local computer for editing.

expression.**CheckOut**(*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 check out.

Remarks

To take advantage of the collaboration features built into Word, documents must be stored on a Microsoft SharePoint Portal Server.

Example

This example verifies that a document is not checked out by another user and that it can be checked out. If the document can be checked out, it copies the document to the local computer for editing.

```
Sub CheckInOut(docCheckOut As String)
    If Documents.CanCheckOut(docCheckOut) = True Then
        Documents.CheckOut docCheckOut
    Else
        MsgBox "You are unable to check out this document at this ti
    End If
End Sub
```

To call the CheckInOut subroutine above, use the following subroutine and replace the "http://servername/workspace/report.doc" file name with an actual file located on a server mentioned in the Remarks section above.

```
Sub CheckDocInOut()
    Call CheckInOut (docCheckIn:="http://servername/workspace/report
End Sub
```



↳ [Show All](#)

CheckSpelling Method

▶ [CheckSpelling method as it applies to the **Application** and **Global** objects.](#)

Checks a string for spelling errors. Returns a **Boolean** to indicate whether the string contains spelling errors. **True** if the string has no spelling errors.

expression.**CheckSpelling**(*Word*, *CustomDictionary*, *IgnoreUppercase*, *MainDictionary*, *CustomDictionary2*, *CustomDictionary3*, *CustomDictionary4*, *CustomDictionary5*, *CustomDictionary6*, *CustomDictionary7*, *CustomDictionary8*, *CustomDictionary9*, *CustomDictionary10*)

expression Required. An expression that returns an **Application** or **Global** object.

Word Required **String**. The text whose spelling is to be checked.

CustomDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the custom dictionary.

IgnoreUppercase Optional **Variant**. **True** if capitalization is ignored. If this argument is omitted, the current value of the **IgnoreUppercase** property is used.

MainDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the main dictionary.

CustomDictionary2 – CustomDictionary10 Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of an additional custom dictionary. You can specify as many as nine additional dictionaries.

▶ [CheckSpelling method as it applies to the **Document** and **Range** objects.](#)

Begins a spelling check for the specified document or range. If the document or range contains errors, this method displays the **Spelling and Grammar** dialog box (**Tools** menu), with the **Check grammar** check box cleared. For a

document, this method checks all available stories (such as headers, footers, and text boxes).

expression.**CheckSpelling**(*CustomDictionary*, *IgnoreUppercase*, *AlwaysSuggest*, *CustomDictionary2*, *CustomDictionary3*, *CustomDictionary4*, *CustomDictionary5*, *CustomDictionary6*, *CustomDictionary7*, *CustomDictionary8*, *CustomDictionary9*, *CustomDictionary10*)

expression Required. An expression that returns a **Document** or **Range** object.

CustomDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the custom dictionary.

IgnoreUppercase Optional **Variant**. **True** if capitalization is ignored. If this argument is omitted, the current value of the **IgnoreUppercase** property is used.

AlwaysSuggest Optional **Variant**. **True** for Microsoft Word to always suggest alternative spellings. If this argument is omitted, the current value of the **SuggestSpellingCorrections** property is used.

CustomDictionary2 – CustomDictionary10 Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of an additional custom dictionary. You can specify as many as nine additional dictionaries.

Example

▶ [As it applies to the **Range** object.](#)

This example begins a spelling check on all available stories of the active document.

```
Set range2 = Documents("MyDocument.doc").Sections(2).Range
range2.CheckSpelling IgnoreUpperCase:=False, _
    CustomDictionary:="MyWork.Dic", _
    CustomDictionary2:="MyTechnical.Dic"
```



CheckSynonyms Method

-

Displays the **Thesaurus** dialog box, which lists alternative word choices, or synonyms, for the text in the specified range.

expression.**CheckSynonyms**

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

Example

This example displays the **Thesaurus** dialog box with synonyms for the selected text.

```
Selection.Range.CheckSynonyms
```

This example displays the **Thesaurus** dialog box with synonyms for the first word in the active document.

```
ActiveDocument.Words(1).CheckSynonyms
```



CleanString Method

-

Removes nonprinting characters (character codes 1 – 29) and special Word characters from the specified string or changes them to spaces (character code 32), as described in the "Remarks" section. Returns the result as a string.

expression.**CleanString**(*String*)

expression Optional. An expression that returns an **Application** object.

String Required **String**. The source string.

Remarks

The following characters are converted as described in this table.

Character code	Description
7 (beep)	Removed unless preceded by character 13 (paragraph), then converted to character 9 (tab).
10 (line feed)	Converted to character 13 (paragraph) unless preceded by character 13, then removed.
13 (paragraph)	Unchanged.
31 (optional hyphen)	Removed.
160 (nonbreaking space)	Converted to character 32 (space).
172 (optional hyphen)	Removed.
176 (nonbreaking space)	Converted to character 32 (space).
182 (paragraph mark)	Removed.
183 (bullet)	Converted to character 32 (space).

Example

This example removes nonprinting characters from the selected text and inserts the result into a new document.

```
Dim strClean As String
Dim docNew As Document

strClean = Application.CleanString(Selection.Text)
Set docNew = Documents.Add
docNew.Content.InsertAfter strClean
```

This example removes nonprinting characters from the selected field code and then displays the result.

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = True
ActiveDocument.Fields(1).Select
MsgBox Application.CleanString(Selection.Text)
```



↳ [Show All](#)

Clear Method

-
DropCap object: Removes the dropped capital letter formatting.

KeyBinding object: Removes the key binding from the **KeyBindings** collection and resets a built-in command to its default key assignment.

ListEntries object: Removes all items from a drop-down form field.

TabStop object: Removes the specified custom tab stop.

TextInput object: Deletes the text from the specified text form field.

expression.**Clear**

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

Example

▶ [As it applies to the **TabStop** object.](#)

This example clears the first custom tab in the first paragraph of the active document.

```
ActiveDocument.Paragraphs(1).TabStops(1).Clear
```

▶ [As it applies to the **TextInput** object.](#)

This example protects the document for forms and deletes the text from the first form field if the field is a text form field.

```
ActiveDocument.Protect Type:=wdAllowOnlyFormFields, NoReset:=True  
If ActiveDocument.FormFields(1).Type = wdFieldFormTextInput Then  
    ActiveDocument.FormFields(1).TextInput.Clear  
End If
```

▶ [As it applies to the **ListEntries** object.](#)

This example removes all items from the form field named "Colors" in Sales.doc.

```
Documents("Sales.doc").FormFields("Colors") _  
    .DropDown.ListEntries.Clear
```

▶ [As it applies to the **DropCap** object.](#)

This example removes dropped capital letter formatting from the first letter in the active document.

```
Set drop = ActiveDocument.Paragraphs(1).DropCap  
If Not (drop Is Nothing) Then drop.Clear
```

▶ [As it applies to the **KeyBinding** object.](#)

This example removes the ALT+F1 key assignment from the Normal template.

```
CustomizationContext = NormalTemplate  
FindKey(BuildKeyCode(Arg1:=wdKeyAlt, Arg2:=wdKeyF1)).Clear
```



↳ [Show All](#)

ClearAll Method

-
TabStops object: Clears all the custom tab stops from the specified paragraphs.

KeyBindings object: Clears all the customized key assignments and restores the original Microsoft Word shortcut key assignments.

Dictionaries or **HangulHanjaConversionDictionaries** object: Unloads all of the custom or conversion dictionaries.

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.

After applying the **ClearAll** method to the **KeyBindings** object, the keys assignments in the specified template or document are reset to the default settings. Use the [CustomizationContext](#) property to specify a document or template context prior to using the **ClearAll** method.

The **ClearAll** method when used on a **Dictionaries** or **HangulHanjaConversionDictionaries** object does not delete the custom or conversion dictionary files. After using this method, the number of custom or conversion dictionaries in the collection is 0 (zero).

Example

▶ [As it applies to the **TabStop** object.](#)

This example clears all the custom tab stops in the active document.

```
ActiveDocument.Paragraphs.TabStops.ClearAll
```

▶ [As it applies to the **KeyBindings** object.](#)

This example clears the customized key assignments in the Normal template. The key assignments are reset to the default settings.

```
CustomizationContext = NormalTemplate  
KeyBindings.ClearAll
```

▶ [As it applies to the **Dictionaries** object.](#)

This example unloads all of the custom dictionaries.

```
CustomDictionaries.ClearAll
```



ClearAllFuzzyOptions Method

-
Clears all nonspecific search options associated with Japanese text.

expression.**ClearAllFuzzyOptions**

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

Remarks

This method sets the following properties to **False**:

[MatchFuzzyAY](#) [MatchFuzzyBV](#)

[MatchFuzzyByte](#)

[MatchFuzzyCase](#)

[MatchFuzzyDash](#)

[MatchFuzzyDZ](#)

[MatchFuzzyHF](#)

[MatchFuzzyHiragana](#)

[MatchFuzzyIterationMark](#)

[MatchFuzzyKanji](#)

[MatchFuzzyKiKu](#)

[MatchFuzzyOldKana](#)

[MatchFuzzyProlongedSoundMark](#)

[MatchFuzzyPunctuation](#)

[MatchFuzzySmallKana](#)

[MatchFuzzySpace](#)

[MatchFuzzyTC](#)

[MatchFuzzyZJ](#)

Example

This example clears all nonspecific search options before executing a search in the selected range. If the word "バイオリン" is formatted as bold, the entire paragraph will be selected and copied to the Clipboard.

```
With Selection.Find
    .ClearFormatting
    .ClearAllFuzzyOptions
    .Font.Bold = True
    .Execute FindText:="バイオリン", Format:=True, Forward:=True
    If .Found = True Then
        .Parent.Expand Unit:=wdParagraph
        .Parent.Copy
    End If
End With
```



↳ [Show All](#)

ClearFormatting Method

-
Removes text and paragraph formatting from a selection or from the formatting specified in a find or replace operation.

expression.**ClearFormatting**

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

Remarks

To ensure that formatting isn't included as criteria in a find or replace operation, use this method before carrying out the operation.

Example

▶ [As it applies to the **Selection** object.](#)

This example removes all text and paragraph formatting from the active document.

```
Sub ClrFmtg()  
  
    ActiveDocument.Select  
    Selection.ClearFormatting  
  
End Sub
```

This example removes all text and paragraph formatting from the second through the fourth paragraphs of the active document.

```
Sub ClrFmtg2()  
  
    ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(2).Range.Start  
        End:=ActiveDocument.Paragraphs(4).Range.End).Select  
    Selection.ClearFormatting  
  
End Sub
```

▶ [As it applies to the **Replacement** object.](#)

This example clears formatting from the find or replace criteria before replacing the word "Inc." with "incorporated" throughout the active document.

```
Sub ClrFmtgReplace()  
  
    Dim rngTemp As Range  
  
    Set rngTemp = ActiveDocument.Content  
    With rngTemp.Find  
        .ClearFormatting  
        .Replacement.ClearFormatting  
        .MatchWholeWord = True  
        .Execute FindText:="Inc.", ReplaceWith:="incorporated", _  
            Replace:=wdReplaceAll  
    End With
```

End Sub

▶ [As it applies to the **Find** object.](#)

This example removes formatting from the find criteria before searching through the selection. If the word "Hello" with bold formatting is found, the entire paragraph is selected and copied to the Clipboard.

```
Sub ClrFmtgFind()  
  
    With Selection.Find  
        .ClearFormatting  
        .Font.Bold = True  
        .Execute FindText:="Hello", Format:=True, Forward:=True  
        If .Found = True Then  
            .Parent.Expand Unit:=wdParagraph  
            .Parent.Copy  
        End If  
    End With  
End Sub
```



↳ [Show All](#)

CloneNode Method

Clones a specified diagram node. Returns a [DiagramNode](#) object that represents the clone.

expression.CloneNode(copyChildren, TargetNode, Pos)

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

copyChildren Required **Boolean**. **True** to clone the diagram node's children as well.

TargetNode Optional **DiagramNode** object. The node where the new node will be placed.

Pos Optional [MsoRelativeNodePosition](#). If *TargetNode* is specified, indicates where the node will be added relative to *TargetNode*.

MsoRelativeNodePosition can be one of these MsoRelativeNodePosition constants.

msoAfterLastSibling

msoAfterNode *default*

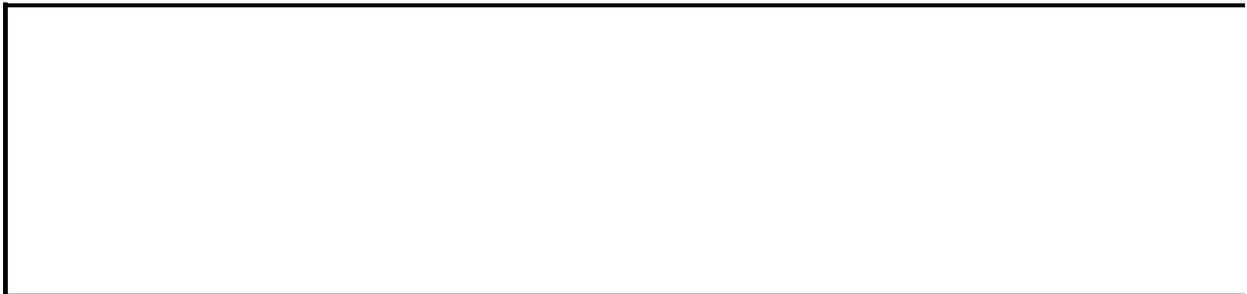
msoBeforeFirstSibling

msoBeforeNode

Example

The following example creates a diagram and clones the most recently created node.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram( _  
        Type:=msoDiagramPyramid, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add child node to the diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    'Apply automatic formatting to the diagram  
    dgnNode.Diagram.AutoFormat = msoTrue  
  
    'Clone the most recently created child node  
    dgnNode.CloneNode CopyChildren:=False  
End Sub
```



↳ [Show All](#)

Close Method

▶ [Close method as it applies to the **Document** and **Documents** objects.](#)

Closes the specified document or documents.

expression.**Close**(*SaveChanges*, *OriginalFormat*, *RouteDocument*)

expression Required. An expression that returns one of the above objects.

SaveChanges Optional **Variant**. Specifies the save action for the document. Can be one of the following **WdSaveOptions** constants: **wdDoNotSaveChanges**, **wdPromptToSaveChanges**, or **wdSaveChanges**.

OriginalFormat Optional **Variant**. Specifies the save format for the document. Can be one of the following **WdOriginalFormat** constants: **wdOriginalDocumentFormat**, **wdPromptUser**, or **wdWordDocument**.

RouteDocument Optional **Variant**. **True** to route the document to the next recipient. If the document doesn't have a routing slip attached, this argument is ignored.

▶ [Close method as it applies to the **MailMergeDataSource**, **Pane**, and **Task** objects.](#)

Closes the specified Mail Merge data source, pane, or task.

expression.**Close**

expression Required. An expression that returns one of the above objects.

▶ [Close method as it applies to the **Window** object.](#)

Closes the specified window.

expression.**Close**(*SaveChanges*, *RouteDocument*)

expression Required. An expression that returns one of the above objects.

SaveChanges Optional **Variant**. Specifies the save action for the document. Can be one of the following **WdSaveOptions** constants: **wdDoNotSaveChanges**, **wdPromptToSaveChanges**, or **wdSaveChanges**.

RouteDocument Optional **Variant**. **True** to route the document to the next recipient. If the document doesn't have a routing slip attached, this argument is ignored.

Example

▶ [As it applies to the **Document** object.](#)

This example prompts the user to save the active document before closing it. If the user clicks **Cancel**, error 4198 (command failed) is trapped and a message is displayed.

```
On Error GoTo errorHandler
ActiveDocument.Close _
    SaveChanges:=wdPromptToSaveChanges, _
    OriginalFormat:=wdPromptUser
errorHandler:
If Err = 4198 Then MsgBox "Document was not closed"
```

▶ [As it applies to the **Pane** object.](#)

This example closes the active pane if the active window is split.

```
If ActiveDocument.ActiveWindow.Panes.Count >= 2 Then _
    ActiveDocument.ActiveWindow.ActivePane.Close
```

▶ [As it applies to the **Task** object.](#)

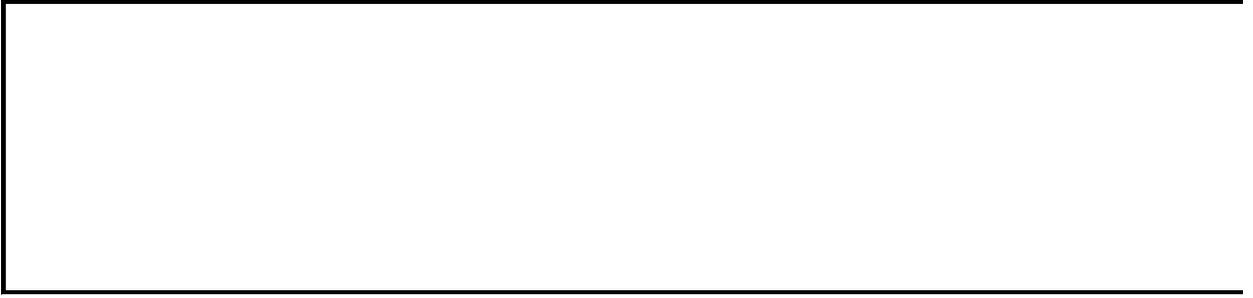
This example activates Microsoft Excel and then closes it.

```
For Each myTask In Tasks
    If InStr(myTask.Name, "Microsoft Excel") > 0 Then
        myTask.Activate
        myTask.Close
    End If
Next myTask
```

▶ [As it applies to the **Window** object.](#)

This example closes the active window of the active document and saves it.

```
ActiveDocument.ActiveWindow.Close SaveChanges:=wdSaveChanges
```



ClosePrintPreview Method

-

Switches the specified document from print preview to the previous view. If the specified document isn't in print preview, an error occurs.

expression.**ClosePrintPreview**

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

Example

This example switches the active window from print preview to normal view.

```
If ActiveDocument.PrintPreview = True Then _  
    ActiveDocument.ClosePrintPreview  
ActiveDocument.ActiveWindow.View.Type = wdNormalView
```



CloseUp Method

-
Removes any spacing before the specified paragraphs.

expression.**CloseUp**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Remarks

The following two statements are equivalent:

```
ActiveDocument.Paragraphs(1).CloseUp  
ActiveDocument.Paragraphs(1).SpaceBefore = 0
```

Example

This example removes any space before the first paragraph in the selection.

```
Selection.Paragraphs(1).CloseUp
```

This example changes the Heading 1 style in the active document so that there's no space before Heading 1 paragraphs.

```
ActiveDocument.Styles("Heading 1").ParagraphFormat.CloseUp
```



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 a **Range** or **Selection** object.

Direction Optional **Variant**. The direction in which to collapse the range or selection. Can be either of the following **WdCollapseDirection** constants: **wdCollapseEnd** or **wdCollapseStart**. The default value is **wdCollapseStart**.

Remarks

If you use **wdCollapseEnd** to collapse a range that refers to an entire paragraph, the range is 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, as shown in the following example.

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.Collapse Direction:=wdCollapseEnd
myRange.MoveEnd Unit:=wdCharacter, Count:=-1
```

Example

This example collapses the selection to an insertion point at the beginning of the previous selection.

```
Selection.Collapse Direction:=wdCollapseStart
```

This example sets myRange equal to the contents of the active document, collapses myRange, and then inserts a 2x2 table at the end of the document.

```
Set myRange = ActiveDocument.Content  
myRange.Collapse Direction:=wdCollapseEnd  
ActiveDocument.Tables.Add Range:=myRange, NumRows:=2, NumColumns:=2
```



CollapseOutline Method

-

Collapses the text under the selection or the specified range by one heading level.

Note If the document isn't in outline or master document view, an error occurs.

expression.CollapseOutline(**Range**)

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

Range Optional **Range** object. The range of paragraphs to be collapsed. If this argument is omitted, the entire selection is collapsed.

Example

This example applies the Heading 2 style to the second paragraph in the active document, switches the active window to outline view, and collapses the text under the second paragraph in the document.

```
ActiveDocument.Paragraphs(2).Style = wdStyleHeading2
With ActiveDocument.ActiveWindow.View
    .Type = wdOutlineView
    .CollapseOutline Range:=ActiveDocument.Paragraphs(2).Range
End With
```

This example collapses every heading in the document by one level.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdOutlineView
    .CollapseOutline Range:=ActiveDocument.Content
End With
```



↳ [Show All](#)

Compare Method

Displays revision marks that indicate where the specified document differs from another document.

expression.Compare(*Name*, *AuthorName*, *CompareTarget*, *DetectFormatChanges*, *IgnoreAllComparisonWarnings*, *AddToRecentFiles*)

expression Required. An expression that returns a [Document](#) object.

Name Required **String**. The name of the document with which the specified document is compared.

AuthorName Optional **Variant**. The reviewer name associated with the differences generated by the comparison. If unspecified, the value defaults to the author name of the revised document or the string "Comparison" if no author information is present.

CompareTarget Optional **Variant**. The target document for the comparison. Can be any [WdCompareTarget](#) constant.

WdCompareTarget can be one of these WdCompareTarget constants.

wdCompareTargetCurrent Places comparison differences in the current document. *Default*.

wdCompareTargetNew Places comparison differences in a new document.

wdCompareTargetSelected Places comparison differences in the target document.

DetectFormatChanges Optional **Boolean**. **True** (default) for the comparison to include detection of format changes.

IgnoreAllComparisonWarnings Optional **Variant**. **True** compares the documents without notifying a user of problems. The default value is **False**.

AddToRecentFiles Optional **Variant**. **True** adds the document to the list of

recently used files on the **File** menu.

Example

This example compares the active document with the document named "FirstRev.doc" in the Draft folder and places the comparison differences in a new document.

```
Sub CompareDocument()  
    ActiveDocument.Compare Name:="C:\Draft\FirstRev.doc", _  
        CompareTarget:=wdCompareTargetNew  
End Sub
```



↳ [Show All](#)

ComputeStatistics Method

▶ [ComputeStatistics method as it applies to the **Range** object.](#)

Returns a statistic based on the contents of the specified range. **Long.**

expression.**ComputeStatistics**(*Statistic*)

expression Required. An expression that returns one of the above objects.

Statistic Required [WdStatistic](#).

WdStatistic can be one of these WdStatistic constants.

wdStatisticCharacters

wdStatisticCharactersWithSpaces

wdStatisticFarEastCharacters

wdStatisticLines

wdStatisticPages

wdStatisticParagraphs

wdStatisticWords

▶ [ComputeStatistics method as it applies to the **Document** object.](#)

Returns a statistic based on the contents of the specified document. **Long.**

expression.**ComputeStatistics**(*Statistic*, *IncludeFootnotesAndEndnotes*)

expression Required. An expression that returns one of the above objects.

Statistic Required [WdStatistic](#).

WdStatistic can be one of these WdStatistic constants.

wdStatisticCharacters
wdStatisticCharactersWithSpaces
wdStatisticFarEastCharacters
wdStatisticLines
wdStatisticPages
wdStatisticParagraphs
wdStatisticWords

IncludeFootnotesAndEndnotes Optional **Variant**. **True** to include footnotes and endnotes when computing statistics. If this argument is omitted, the default value is **False**.

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

▶ [As it applies to the **Range** object.](#)

This example displays the number of words and characters in the first paragraph of Report.doc.

```
Set myRange = Documents("Report.doc").Paragraphs(1).Range
wordCount = myRange.ComputeStatistics(Statistic:=wdStatisticWords)
charCount = _
    myRange.ComputeStatistics(Statistic:=wdStatisticCharacters)
MsgBox "The first paragraph contains " & wordCount _
    & " words and a total of " & charCount & " characters."
```

▶ [As it applies to the **Document** object.](#)

This example displays the number of words in the active document, including footnotes.

```
MsgBox _
    ActiveDocument.ComputeStatistics(Statistic:=wdStatisticWords, _
    IncludeFootnotesAndEndnotes:=True) & " words"
```



↳ [Show All](#)

Condition Method

Returns a [ConditionalStyle](#) object that represents special style formatting for a portion of a table.

expression.**Condition**(*ConditionCode*)

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

ConditionCode Required [WdConditionCode](#). The are of the table to which to apply the formatting.

WdConditionCode can be one of these WdConditionCode constants.

wdEvenColumnBanding Applies formatting to even-numbered columns.

wdEvenRowBanding Applies formatting to even-numbered rows.

wdFirstColumn Applies formatting to the first column in a table.

wdFirstRow Applies formatting to the first row in a table.

wdLastColumn Applies formatting to the last column in a table.

wdLastRow Applies formatting to the last row in a table.

wdNECell Applies formatting to the last cell in the first row.

wdNWCell Applies formatting to the first cell in the first row.

wdOddColumnBanding Applies formatting to odd-numbered columns.

wdOddRowBanding Applies formatting to odd-numbered rows.

wdSECell Applies formatting to the last cell in the table.

wdSWCell Applies formatting to first cell in the last row of the table.

Example

This example selects the first table in the active document and adds a 20 percent shading to odd-numbered columns.

```
Sub TableStylesTest()  
  With ActiveDocument  
  
    'Select the table to which the conditional  
    'formatting will apply  
    .Tables(1).Select  
  
    'Specify the conditional formatting  
    .Styles("Table Grid").Table _  
      .Condition(wdOddColumnBanding).Shading _  
      .BackgroundPatternColor = wdColorGray20  
  End With  
End Sub
```



Connect Method

-
Establishes a connection to a network drive.

expression.**Connect**(*Path*, *Drive*, *Password*)

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

Path Required **String**. The path for the network drive (for example, "\\Project\Info").

Drive Optional **Variant**. A number corresponding to the letter you want to assign to the network drive, where 0 (zero) corresponds to the first available drive letter, 1 corresponds to the second available drive letter, and so on. If this argument is omitted, the next available letter is used.

Password Optional **Variant**. The password, if the network drive is protected with a password.

Remarks

Use the [Dialogs](#) property with the **wdDialogConnect** constant to display the **Connect To Network Drive** dialog box. The following example displays the **Connect To Network Drive** dialog box, with a preset path shown.

```
With Dialogs(wdDialogConnect)
    .Path = "\\Marketing\Public"
    .Show
End With
```

Example

This example establishes a connection to a network drive (\\Project\Info) protected with the password "smiley" and assigns the network drive to the next available drive letter.

```
System.Connect Path:="\\Project\Info", Password:="smiley"
```

This example establishes a connection to a network drive (\\Team1\Public) and assigns the network drive to the third available drive letter.

```
System.Connect Path:="\\Team1\Public", Drive:=2
```



↳ [Show All](#)

Convert Method

▶ [Convert method as it applies to the **Diagram** object.](#)

Converts a diagram of one type into a diagram of another type.

expression.**Convert**(*Type*)

expression Required. An expression that returns a [Diagram](#) object.

Type Required [MsoDiagramType](#). The type of diagram to which to convert.

MsoDiagramType can be one of these MsoDiagramType constants.

msoDiagramCycle Shows a process with a continuous cycle.

msoDiagramMixed Not used with this method.

msoDiagramOrgChart Shows hierarchical relationships.

msoDiagramPyramid Shows foundation-based relationships.

msoDiagramRadial Shows relationships of a core element.

msoDiagramTarget Shows steps toward a goal.

msoDiagramVenn Shows areas of overlap between elements.

▶ [Convert method as it applies to the **Endnotes** and **Footnotes** objects.](#)

Converts endnotes to footnotes, or vice versa.

expression.**Convert**

expression Required. An expression that returns one of the above objects.

▶ [Convert method as it applies to the **ListTemplate** object.](#)

Converts a multiple-level list to a single-level list, or vice versa.

expression.**Convert**(*Level*)

expression Required. An expression that returns a [ListTemplate](#) object.

Level Optional **Variant**. The level to use for formatting the new list. When converting a multiple-level list to a single-level list, this argument can be a number from 1 through 9. When converting a single-level list to a multiple-level list, 1 is the only valid value. If this argument is omitted, 1 is the default value.

Remarks

You cannot use the **Convert** method on a list template that is derived from the [ListGalleries](#) collection.

Example

▶ [As it applies to the **Diagram** object.](#)

This example creates a pyramid diagram and then converts it into a radial diagram.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram( _  
        Type:=msoDiagramPyramid, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add four child nodes to the diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    With dgnNode.Diagram  
  
        'Automatically formats the diagram  
        .AutoFormat = msoTrue  
  
        'Converts the diagram from a pyramid to a radial diagram  
        .Convert Type:=msoDiagramRadial  
    End With  
  
End Sub
```

▶ [As it applies to the **Endnotes** object.](#)

This example converts all endnotes in the active document to footnotes.

```
Set endDocEndnotes = ActiveDocument.Endnotes  
If endDocEndnotes.Count > 0 Then myEndnotes.Convert
```

▶ [As it applies to the **Footnotes** object.](#)

This example converts the footnotes in the selection to endnotes.

```
If Selection.Footnotes.Count > 0 Then Selection.Footnotes.Convert
```

► [As it applies to the **ListTemplate** object.](#)

This example converts the first list template in the active document. If the list template is multiple-level, it becomes single-level, or vice versa.

```
ActiveDocument.ListTemplates(1).Convert
```



ConvertHangulAndHanja Method

Converts the specified range from hangul to hanja or vice versa.

expression.**ConvertHangulAndHanja**(*ConversionsMode*, *FastConversion*, *CheckHangulEnding*, *EnableRecentOrdering*, *CustomDictionary*)

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

ConversionsMode Optional **Variant**. Sets the direction for the conversion between hangul and hanja. Can be either of the following **WdMultipleWordConversionsMode** constants: **wdHangulToHanja** or **wdHanjaToHangul**. The default value is the current value of the [MultipleWordConversionsMode](#) property.

FastConversion Optional **Variant**. **True** if Microsoft Word automatically converts a word with only one suggestion for conversion. The default value is the current value of the [HangulHanjaFastConversion](#) property.

CheckHangulEnding Optional **Variant**. **True** if Word automatically detects hangul endings and ignores them. The default value is the current value of the [CheckHangulEndings](#) property. This argument is ignored if the **ConversionsMode** argument is set to **wdHanjaToHangul**.

EnableRecentOrdering Optional **Variant**. **True** if Word displays the most recently used words at the top of the suggestions list. The default value is the current value of the [EnableHangulHanjaRecentOrdering](#) property.

CustomDictionary Optional **Variant**. The name of a custom hangul-hanja conversion dictionary. Use this argument in order to use a custom dictionary with hangul-hanja conversions not contained in the main dictionary.

Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example converts the current selection from hangul to hanja.

```
Selection.Range.ConvertHangulAndHanja _  
    ConversionsMode:=wdHangulToHanja, _  
    FastConversion:=True, _  
    EnableRecentOrdering:= True
```



↳ [Show All](#)

ConvertNumbersToText Method

Changes the list numbers and LISTNUM fields in the specified **Document**, **List**, or **ListFormat** object to text.

expression.**ConvertNumbersToText**(*NumberType*)

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

NumberType Optional **Variant**. The type of number to be converted. Can be any of the following [WdNumberType](#) constant.

WdNumberType can be one of these WdNumberType constants.

wdNumberParagraph

wdNumberListNum Default value for LISTNUM fields.

wdNumberAllNumbers Default value for all other cases.

Remarks

There are two types of numbers: preset numbers (**wdNumberParagraph**), which you can add to paragraphs by selecting a template in the **Bullets and Numbering** dialog box; and LISTNUM fields (**wdNumberListNum**), which allow you to add more than one number per paragraph.

The **ConvertNumbersToText** method is useful if you want to work with a document in another application and that application doesn't recognize list formatting or LISTNUM fields.

After you've converted list numbers to text, you can no longer manipulate them in a list.

Example

▶ [As it applies to the **Document** object.](#)

This example converts the list numbers and LISTNUM fields in the active document to text.

```
ActiveDocument.ConvertNumbersToText
```

▶ [As it applies to the **List** object.](#)

This example converts the numbers in the first list to text.

```
ActiveDocument.Lists(1).ConvertNumbersToText
```

▶ [As it applies to the **ListFormat** object.](#)

This example converts the preset numbers in myRange to text without affecting any LISTNUM fields.

```
Set myDoc = ActiveDocument  
Set myRange = _  
    myDoc.Range(Start:=myDoc.Paragraphs(12).Range.Start, _  
    End:=myDoc.Paragraphs(20).Range.End)  
myRange.ListFormat.ConvertNumbersToText wdNumberParagraph
```



ConvertTo Method

Converts the specified OLE object from one class to another, making it possible for you to edit the object in a different server application, or changing how the object is displayed in the document.

expression.**ConvertTo**(*ClassType*, *DisplayAsIcon*, *IconFileName*, *IconIndex*, *IconLabel*)

expression Required. An expression that returns an **OLEFormat** object.

ClassType Optional **Variant**. The name of the application used to activate the OLE object. You can see a list of the available applications in the **Object type** box on the **Create New** tab in the **Object** dialog box (**Insert** menu). You can find the **ClassType** string by inserting an object as an inline shape and then viewing the field codes. The class type of the object follows either the word "EMBED" or the word "LINK."

DisplayAsIcon Optional **Variant**. **True** to display the OLE object as an icon. The default value is **False**.

IconFileName Optional **Variant**. The file that contains the icon to be displayed.

IconIndex Optional **Variant**. The index number of the icon within **IconFileName**. The order of icons in the specified file corresponds to the order in which the icons appear in the **Change Icon** dialog box (**Insert** menu, **Object** dialog box) when the **Display as icon** check box is selected. The first icon in the file has the index number 0 (zero). If an icon with the given index number doesn't exist in **IconFileName**, the icon with the index number 1 (the second icon in the file) is used. The default value is 0 (zero).

IconLabel Optional **Variant**. A label (caption) to be displayed beneath the icon.

Example

This example creates a new document, then inserts an embedded Word document with some text. Then, the embedded document is converted to a Word Picture.

```
Dim objEmbedded As Object

Documents.Add

Set objEmbedded = ActiveDocument.Shapes _
    .AddOLEObject(ClassType:= "Word.Document")
objEmbedded.Activate
Selection.TypeText "Test"
objEmbedded.OLEFormat.OLEFormat.ConvertTo _
    ClassType:="Word.Picture"
```



ConvertToFrame Method

-

Converts the specified shape to a frame. Returns a [Frame](#) object that represents the new frame.

expression.**ConvertToFrame**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Remarks

Shapes that don't support attached text cannot be converted to frames. For pictures, OLE objects, and ActiveX controls, use the [ConvertToInlineShape](#) method.

If you use this method on a **ShapeRange** object that contains more than one shape, an error occurs.

In Word 97 and later, frames have been replaced by text boxes.

Example

This example creates a text box using the selected text, and then it converts the text box to a frame.

```
If Selection.Type = wdSelectionNormal Then
    Selection.CreateTextbox
    Selection.ShapeRange.ConvertToFrame
End If
```



ConvertToInlineShape Method

-

Converts the specified shape in the drawing layer of a document to an inline shape in the text layer. You can convert only shapes that represent pictures, OLE objects, or ActiveX controls. This method returns an [InlineShape](#) object that represents the picture or OLE object.

expression.**ConvertToInlineShape**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Remarks

Shapes that support attached text cannot be converted to inline shapes. For these shapes, use the [ConvertToFrame](#) method.

If you use this method on a **ShapeRange** object that contains more than one shape, an error occurs.

Example

This example converts each picture in MyDoc.doc to an inline shape.

```
For Each s In Documents("MyDoc.doc").Shapes
    If s.Type = msoPicture Then
        s.ConvertToInlineShape
    End If
Next s
```



↳ [Show All](#)

ConvertToShape Method

▶ [ConvertToShape method as it applies to the **FreeformBuilder** object.](#)

Creates a shape that has the geometric characteristics of the specified object. Returns a **Shape** object that represents the new shape.

expression.**ConvertToShape**(*Anchor*)

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

Anchor Optional **Variant**. A **Range** object that represents the text to which the shape is bound. If **Anchor** is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, the anchoring range is selected automatically and the shape is positioned relative to the top and left edges of the page.

▶ [ConvertToShape method as it applies to the **InlineShape** object.](#)

Converts an inline shape to a free-floating shape. Returns a **Shape** object that represents the new shape.

expression.**ConvertToShape**

expression Required. An expression that returns an **InlineShapes** object.

Remarks

You must apply the [AddNodes](#) method to a **FreeformBuilder** object at least once before you use the **ConvertToShape** method.

Example

▶ [As applies to the **InlineShape** object.](#)

This example converts the first inline shape in the active document to a floating shape.

```
ActiveDocument.InlineShapes(1).ConvertToShape
```

▶ [As applies to the **FreeFormBuilder** object.](#)

This example adds a freeform with five vertices to myDocument.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.BuildFreeform(msoEditingCorner, 360, 200)
    .AddNodes msoSegmentCurve, msoEditingCorner, _
        380, 230, 400, 250, 450, 300
    .AddNodes msoSegmentCurve, msoEditingAuto, 480, 200
    .AddNodes msoSegmentLine, msoEditingAuto, 480, 400
    .AddNodes msoSegmentLine, msoEditingAuto, 360, 200
    .ConvertToShape
End With
```



↳ [Show All](#)

ConvertToTable Method

Converts text within a range or selection to a table. Returns the table as a **Table** object.

expression.**ConvertToTable**(*Separator, NumRows, NumColumns, InitialColumnWidth, Format, ApplyBorders, ApplyShading, ApplyFont, ApplyColor, ApplyHeadingRows, ApplyLastRow, ApplyFirstColumn, ApplyLastColumn, AutoFit, AutoFitBehavior, DefaultTableBehavior*)

expression Required. An expression that returns a **Range** or **Selection** object.

Separator Optional **Variant**. Specifies the character used to separate text into cells. Can be a character or one of the following [WdTableFieldSeparator](#) constant. If this argument is omitted, the value of the **DefaultTableSeparator** property is used.

WdTableFieldSeparator can be one of these WdTableFieldSeparator constants.

wdSeparateByCommas

wdSeparateByDefaultListSeparator

wdSeparateByParagraphs

wdSeparateByTabs

NumRows Optional **Variant**. The number of rows in the table. If this argument is omitted, Microsoft Word sets the number of rows, based on the contents of the range or selection.

NumColumns Optional **Variant**. The number of columns in the table. If this argument is omitted, Word sets the number of columns, based on the contents of the range or selection.

InitialColumnWidth Optional **Variant**. The initial width of each column, in points. If this argument is omitted, Word calculates and adjusts the column width so that the table stretches from margin to margin.

Format Optional **Variant**. Specifies one of the predefined formats listed in the **Table AutoFormat** dialog box (**Table** menu). Can be one of the [WdTableFormat](#) constants.

Can be one of the following WdTableFormat constants:

wdTableFormat3DEffects1

wdTableFormat3DEffects2

wdTableFormat3DEffects3

wdTableFormatClassic1

wdTableFormatClassic2

wdTableFormatClassic3

wdTableFormatClassic4

wdTableFormatColorful1

wdTableFormatColorful2

wdTableFormatColorful3

wdTableFormatColumns1

wdTableFormatColumns2

wdTableFormatColumns3

wdTableFormatColumns4

wdTableFormatColumns5

wdTableFormatContemporary

wdTableFormatElegant

wdTableFormatGrid1

wdTableFormatGrid2

wdTableFormatGrid3

wdTableFormatGrid4

wdTableFormatGrid5

wdTableFormatGrid6

wdTableFormatGrid7

wdTableFormatGrid8

wdTableFormatList1

wdTableFormatList2

wdTableFormatList3

wdTableFormatList4

wdTableFormatList5
wdTableFormatList6
wdTableFormatList7
wdTableFormatList8
wdTableFormatNone
wdTableFormatProfessional
wdTableFormatSimple1
wdTableFormatSimple2
wdTableFormatSimple3
wdTableFormatSubtle1
wdTableFormatSubtle2
wdTableFormatWeb1
wdTableFormatWeb2
wdTableFormatWeb3

ApplyBorders Optional **Variant**. **True** to apply the border properties of the specified format.

ApplyShading Optional **Variant**. **True** to apply the shading properties of the specified format.

ApplyFont Optional **Variant**. **True** to apply the font properties of the specified format.

ApplyColor Optional **Variant**. **True** to apply the color properties of the specified format.

ApplyHeadingRows Optional **Variant**. **True** to apply the heading-row properties of the specified format.

ApplyLastRow Optional **Variant**. **True** to apply the last-row properties of the specified format.

ApplyFirstColumn Optional **Variant**. **True** to apply the first-column properties of the specified format.

ApplyLastColumn Optional **Variant**. **True** to apply the last-column properties

of the specified format.

AutoFit Optional **Variant**. **True** to decrease the width of the table columns as much as possible without changing the way text wraps in the cells.

AutoFitBehavior Optional **Variant**. Sets the AutoFit rules for how Word sizes a table. Can be one of the following [WdAutoFitBehavior](#) constant. If **DefaultTableBehavior** is **wdWord8TableBehavior**, this argument is ignored.

WdAutoFitBehavior can be one of these WdAutoFitBehavior constants.

wdAutoFitContent

wdAutoFitFixed

wdAutoFitWindow

DefaultTableBehavior Optional **Variant**. Sets a value that specifies whether Microsoft Word automatically resizes cells in a table to fit the contents (AutoFit). Can be one of the following [WdDefaultTableBehavior](#) constant.

WdDefaultTableBehavior can be one of these WdDefaultTableBehavior constants.

wdWord8TableBehavior Disables AutoFit. Default.

wdWord9TableBehavior Enables AutoFit.

Example

▶ [As it applies to the **Range** object.](#)

This example converts the first three paragraphs in the active document to a table.

```
Set aDoc = ActiveDocument
Set myRange = aDoc.Range(Start:=aDoc.Paragraphs(1).Range.Start, _
    End:=aDoc.Paragraphs(3).Range.End)
myRange.ConvertToTable Separator:=wdSeparateByParagraphs
```

▶ [As it applies to the **Selection** object.](#)

This example inserts text at the insertion point and then converts the comma-delimited text to a table with formatting.

```
With Selection
    .Collapse
    .InsertBefore "one, two, three"
    .InsertParagraphAfter
    .InsertAfter "one, two, three"
    .InsertParagraphAfter
End With
Set myTable = _
    Selection.ConvertToTable(Separator:=wdSeparateByCommas, _
        Format:=wdTableFormatList8)
```



ConvertToText Method

Converts a table to text and returns a **Range** object that represents the delimited text.

expression.**ConvertToText**(*Separator*, *NestedTables*)

expression Required. An expression that returns a **Row**, **Rows**, or **Table** object.

Separator Optional **Variant**. The character that delimits the converted columns (paragraph marks delimit the converted rows). Can be any following [WdTableFieldSeparator](#) constants..

WdTableFieldSeparator can be one of these WdTableFieldSeparator constants.

wdSeparateByCommas

wdSeparateByDefaultListSeparator

wdSeparateByParagraphs

wdSeparateByTabs Default.

NestedTables Optional **Variant**. **True** if nested tables are converted to text. This argument is ignored if **Separator** is not **wdSeparateByParagraphs**. The default value is **True**.

Remarks

When you apply the **ConvertToText** method to a **Table** object, the object is deleted. To maintain a reference to the converted contents of the table, you must assign the **Range** object returned by the **ConvertToText** method to a new object variable. In the following example, the first table in the active document is converted to text and then formatted as a bulleted list.

```
Dim tableTemp As Table
Dim rngTemp As Range

Set tableTemp = ActiveDocument.Tables(1)
Set rngTemp = _
    tableTemp.ConvertToText(Separator:=wdSeparateByParagraphs)

rngTemp.ListFormat.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdBulletGallery).ListTemplates(1)
```

Example

This example creates a table and then converts it to text by using tabs as separator characters.

```
Dim docNew As Document
Dim tableNew As Table
Dim intTemp As Integer
Dim cellLoop As Cell
Dim rngTemp As Range

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=3)

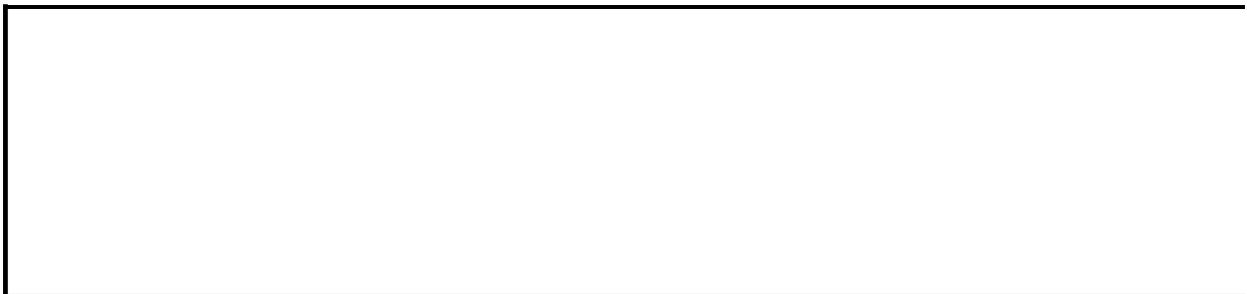
intTemp = 1

For Each cellLoop In tableNew.Range.Cells
    cellLoop.Range.InsertAfter "Cell " & intTemp
    intTemp = intTemp + 1
Next cellLoop

MsgBox "Click OK to convert table to text."
Set rngTemp = _
    tableNew.ConvertToText(Separator:=wdSeparateByTabs)
```

This example converts the table that contains the selection to text, with spaces between the columns.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).ConvertToText Separator:=" "
Else
    MsgBox "The insertion point is not in a table."
End If
```



ConvertVietDoc Method

-
Reconverts a Vietnamese document to Unicode using a code page other than the default.

expression.ConvertVietDoc(*CodePageOrigin*)

expression Required. An expression that returns a [Document](#) object.

CodePageOrigin Required **Long**. The original code page used to encode the document.

Remarks

Use the **ConvertVietDoc** method if you want a document to be viewable on another computer or platform.

Example

This example converts the active document from the Vietnamese ABC code page to Unicode. This example assumes that the active document is encoded using the Vietnamese ABC code page.

```
Sub ConvertToVietCodePage()  
    ActiveDocument.ConvertVietDoc CodePageOrigin:=5  
End Sub
```



↳ [Show All](#)

Copy Method

▶ [Copy method as it applies to the **Bookmark** object.](#)

Sets the bookmark specified by the *Name* argument to the location marked by another bookmark, and returns a **Bookmark** object. **Bookmark** object.

expression.**Copy**(*Name*)

expression Required. An expression that returns one of the above objects.

Name Required **String**. The name of the new bookmark.

▶ [Copy method as it applies to the **Field, FormField, Frame, MailMergeField, PageNumber, Range, and Selection** objects.](#)

Copies the specified object to the Clipboard.

expression.**Copy**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Selection** object.](#)

This example copies the contents of the selection into a new document.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Copy
    Documents.Add.Content.Paste
End If
```

▶ [As it applies to the **BookMark** object.](#)

This example sets the Book2 bookmark to the location marked by the Book1 bookmark.

```
ActiveDocument.Bookmarks("Book1").Copy Name:="Book2"
```

▶ [As it applies to the **Range** object.](#)

This example sets the Selection bookmark to the \Sel predefined bookmark in the active document.

```
ActiveDocument.Bookmarks("\Sel").Copy Name:="Selection"
```

This example copies the first paragraph in the active document and pastes it at the end of the document.

```
ActiveDocument.Paragraphs(1).Range.Copy
Set myRange = ActiveDocument.Range _
    (Start:=ActiveDocument.Content.End - 1, _
    End:=ActiveDocument.Content.End - 1)
myRange.Paste
```

This example copies the comments in the active document to the Clipboard.

```
If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.StoryRanges(wdCommentsStory).Copy
End If
```



CopyAsPicture Method

The **CopyAsPicture** method works the same way as the [Copy](#) method for **Range** and **Selection** objects.

expression.**CopyAsPicture**

expression Required. An expression that returns a **Range** or **Selection** object.

Example

This example copies the contents of the active document as a picture and pastes it as a picture at the end of the document.

```
Sub CopyPasteAsPicture()  
    ActiveDocument.Content.Select  
    With Selection  
        .CopyAsPicture  
        .Collapse Direction:=wdCollapseEnd  
        .PasteSpecial DataType:=wdPasteMetafilePicture  
    End With  
End Sub
```



CopyFormat Method

-

Copies the character formatting of the first character in the selected text. If a paragraph mark is selected, Word copies paragraph formatting in addition to character formatting.

Note You can apply the copied formatting to another selection by using the **PasteFormat** method.

expression.**CopyFormat**

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

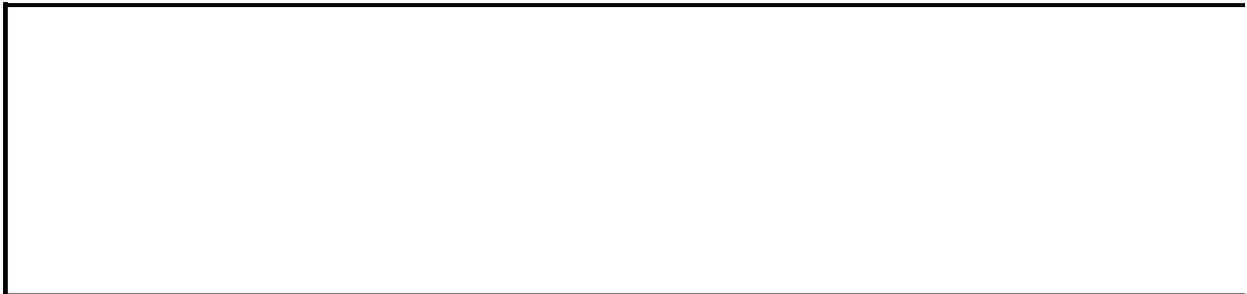
Example

This example copies the formatting of the first paragraph to the second paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Range.Select  
Selection.CopyFormat  
ActiveDocument.Paragraphs(2).Range.Select  
Selection.PasteFormat
```

This example collapses the selection and copies its character formatting to the next word.

```
With Selection  
    .Collapse Direction:=wdCollapseStart  
    .CopyFormat  
    .Next(Unit:=wdWord, Count:=1).Select  
    .PasteFormat  
End With
```



CopyStylesFromTemplate Method

-
Copies styles from the specified template to a document.

expression.**CopyStylesFromTemplate**(*Template*)

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

Template Required **String**. The template file name.

Remarks

When styles are copied from a template to a document, like-named styles in the document are redefined to match the style descriptions in the template. Unique styles from the template are copied to the document. Unique styles in the document remain intact.

Example

This example copies the styles from the active document's template to the document.

```
ActiveDocument.CopyStylesFromTemplate _  
    Template:=ActiveDocument.AttachedTemplate.FullName
```

This example copies the styles from the Sales96.dot template to Sales.doc.

```
Documents("Sales.doc").CopyStylesFromTemplate _  
    Template:="C:\MSOffice\Templates\Sales96.dot"
```



↳ [Show All](#)

CountNumberedItems Method

Returns the number of bulleted or numbered items and LISTNUM fields in the specified **Document**, **List**, or **ListFormat** object.

expression.CountNumberedItems(NumberType, Level)

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

NumberType Optional **Variant**. The type of numbers to be counted. Can be one of the following **WdNumberType** constants: **wdNumberParagraph**, **wdNumberListNum**, or **wdNumberAllNumbers**. The default value is **wdNumberAllNumbers**.

Level Optional **Variant**. A number that corresponds to the numbering level you want to count. If this argument is omitted, all levels are counted.

Remarks

Bulleted items are counted when either **wdNumberParagraph** or **wdNumberAllNumbers** (the default) is specified for *NumberType*.

There are two types of numbers: preset numbers (**wdNumberParagraph**), which you can add to paragraphs by selecting a template in the **Bullets and Numbering** dialog box; and LISTNUM fields (**wdNumberListNum**), which allow you to add more than one number per paragraph.

Example

▶ [As applies to the **ListFormat** object.](#)

This example formats the current selection as a list, using the second numbered list template. The example then counts the numbered and bulleted items and LISTNUM fields in the active document and displays the result in a message box.

```
Selection.Range.ListFormat.ApplyListTemplate _  
    ListTemplate:=ListGalleries(wdNumberGallery).ListTemplates(2)  
Msgbox ActiveDocument.CountNumberedItems
```

This example counts the number of first-level numbered or bulleted items in the active document.

```
Msgbox ActiveDocument.Content.ListFormat _  
    .CountNumberedItems(Level:=1)
```

This example counts the number of LISTNUM fields in the variable myRange. The result is displayed in a message box.

```
Set myDoc = ActiveDocument  
Set myRange = _  
    myDoc.Range(Start:=myDoc.Paragraphs(12).Range.Start, _  
    End:=myDoc.Paragraphs(50).Range.End)  
numfields = myRange.ListFormat.CountNumberedItems(wdNumberListNum)  
Msgbox numfields
```

▶ [As applies to the **List** object.](#)

This example displays a message box that reports the number of items in each list in MyLetter.

```
i = 1
Set myDoc = Documents("MyLetter.doc")
For Each li In myDoc.Lists
    MsgBox "List " & i & " has " _
        & li.CountNumberedItems & " items."
    i = i + 1
Next li
```

CreateAutoTextEntry Method

-

Adds a new [AutoTextEntry](#) object to the [AutoTextEntries](#) collection, based on the current selection.

expression.CreateAutoTextEntry(*Name*, *StyleName*)

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

Name Required **String**. The text the user must type to call the new AutoText entry.

StyleName Required **String**. The category in which the new AutoText entry will be listed on the **AutoText** menu.

Example

This example creates a new `AutoText` entry named "handdel" under the category "Mailing Instructions," given "HAND DELIVERY" as the currently selected text.

```
Selection.CreateAutoTextEntry "handdel", _  
    "Mailing Instructions"
```



CreateDataSource Method

Creates a Word document that uses a table to store data for a mail merge. The new data source is attached to the specified document, which becomes a main document if it's not one already.

expression.**CreateDataSource**(*Name*, *PasswordDocument*, *WritePasswordDocument*, *HeaderRecord*, *MSQuery*, *SQLStatement*, *SQLStatement1*, *Connection*, *LinkToSource*)

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

Name Optional **Variant**. The path and file name for the new data source.

PasswordDocument Optional **Variant**. The password required to open the new data source.

WritePasswordDocument Optional **Variant**. The password required to save changes to the data source.

HeaderRecord Optional **Variant**. Field names for the header record. If this argument is omitted, the standard header record is used: "Title, FirstName, LastName, JobTitle, Company, Address1, Address2, City, State, PostalCode, Country, HomePhone, WorkPhone." To separate field names, use the list separator specified in **Regional Settings** in **Control Panel**.

MSQuery Optional **Variant**. **True** to launch Microsoft Query, if it's installed. The **FileName**, **PasswordDoc**, and **HeaderRecord** arguments are ignored.

SQLStatement Optional **Variant**. Defines query options for retrieving data.

SQLStatement1 Optional **Variant**. If the query string is longer than 255 characters, **SQLStatement** specifies the first portion of the string, and **SQLStatement1** specifies the second portion.

Connection Optional **Variant**. A range within which the query specified by

SQLStatement will be performed. How you specify the range depends on how data is retrieved. For example:

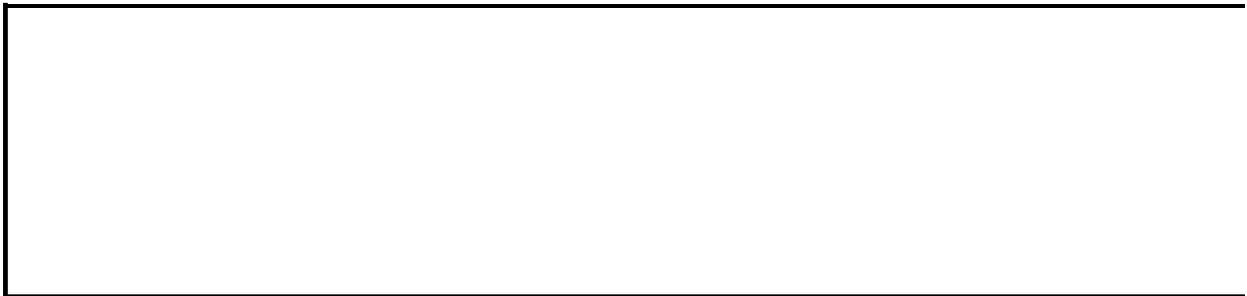
- When retrieving data through ODBC, you specify a connection string.
- When retrieving data from Microsoft Excel using dynamic data exchange (DDE), you specify a named range.
- When retrieving data from Microsoft Access, you specify the word "Table" or "Query" followed by the name of a table or query.

LinkToSource Optional **Variant**. **True** to perform the query specified by **Connection** and **SQLStatement** each time the main document is opened.

Example

This example creates a new data source document named "Data.doc" and attaches the data source to the active document. The new data source includes a five-column table that has the field names specified by the ***HeaderRecord*** argument.

```
ActiveDocument.MailMerge.CreateDataSource _  
    Name:="C:\Documents\Data.doc", _  
    HeaderRecord:="Name, Address, City, State, Zip"
```



CreateHeaderSource Method

Creates a Word document that stores a header record that's used in place of the data source header record in a mail merge. This method attaches the new header source to the specified document, which becomes a main document if it's not one already.

Note The new header source uses a table to arrange mail merge field names.

expression.**CreateHeaderSource**(*Name*, *PasswordDocument*, *WritePasswordDocument*, *HeaderRecord*)

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

Name Required **String**. The path and file name for the new header source.

PasswordDocument Optional **Variant**. The password required to open the new header source.

WritePasswordDocument Optional **Variant**. The password required to save changes to the header source.

HeaderRecord Optional **Variant**. A string that specifies the field names for the header record. If this argument is omitted, the standard header record is used: "Title, FirstName, LastName, JobTitle, Company, Address1, Address2, City, State, PostalCode, Country, HomePhone, WorkPhone." To separate field names in Windows, use the list separator specified in **Regional Settings** in **Control Panel**.

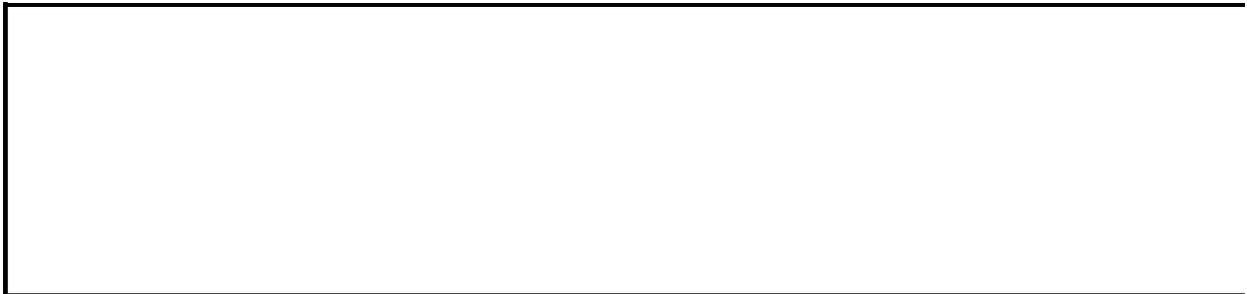
Example

This example creates a header source with five field names and attaches the new header source named "Header.doc" to the active document.

```
ActiveDocument.MailMerge.CreateHeaderSource Name:="Header.doc", _  
    HeaderRecord:="Name, Address, City, State, Zip"
```

This example creates a header source for the document named "Main.doc" (with the standard header record) and opens the data source named "Data.doc."

```
With Documents("Main.doc").MailMerge  
    .CreateHeaderSource Name:="Fields.doc"  
    .OpenDataSource Name:="C:\Documents\Data.doc"  
End With
```



CreateLetterContent Method

Creates and returns a **LetterContent** object based on the specified letter elements. **LetterContent** object.

expression.CreateLetterContent(*DateFormat*, *IncludeHeaderFooter*, *PageDesign*, *LetterStyle*, *Letterhead*, *LetterheadLocation*, *LetterheadSize*, *RecipientName*, *RecipientAddress*, *Salutation*, *SalutationType*, *RecipientReference*, *MailingInstructions*, *AttentionLine*, *Subject*, *CCList*, *ReturnAddress*, *SenderName*, *Closing*, *SenderCompany*, *SenderJobTitle*, *SenderInitials*, *EnclosureNumber*, *InfoBlock*, *RecipientCode*, *RecipientGender*, *ReturnAddressShortForm*, *SenderCity*, *SenderCode*, *SenderGender*, *SenderReference*)

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

DateFormat Required **String**. The date for the letter.

IncludeHeaderFooter Required **Boolean**. **True** to include the header and footer from the page design template.

PageDesign Required **String**. The name of the template attached to the document.

LetterStyle Required [WdLetterStyle](#). The document layout.

WdLetterStyle can be one of these WdLetterStyle constants.

wdFullBlock

wdModifiedBlock

wdSemiBlock

Letterhead Required **Boolean**. **True** to reserve space for a preprinted letterhead.

LetterheadLocation Required [WdLetterheadLocation](#). The location of the

preprinted letterhead.

WdLetterheadLocation can be one of these WdLetterheadLocation constants.

wdLetterBottom

wdLetterLeft

wdLetterRight

wdLetterTop

LetterheadSize Required **Single**. The amount of space (in points) to be reserved for a preprinted letterhead.

RecipientName Required **String**. The name of the person who'll be receiving the letter.

RecipientAddress Required **String**. The mailing address of the person who'll be receiving the letter.

Salutation Required **String**. The salutation text for the letter.

SalutationType Required [WdSalutationType](#). The salutation type for the letter.

WdSalutationType can be one of these WdSalutationType constants.

wdSalutationFormal

wdSalutationOther

wdSalutationBusiness

wdSalutationInformal

RecipientReference Required **String**. The reference line text for the letter (for example, "In reply to:").

MailingInstructions Required **String**. The mailing instruction text for the letter (for example, "Certified Mail").

AttentionLine Required **String**. The attention line text for the letter (for example, "Attention:").

Subject Required **String**. The subject text for the specified letter.

CCList Required **String**. The names of the carbon copy (CC) recipients for the letter.

ReturnAddress Required **String**. The text of the return mailing address for the letter.

SenderName Required **String**. The name of the person sending the letter.

Closing Required **String**. The closing text for the letter.

SenderCompany Required **String**. The company name of the person creating the letter.

SenderJobTitle Required **String**. The job title of the person creating the letter.

SenderInitials Required **String**. The initials of the person creating the letter.

EnclosureNumber Required **Long**. The number of enclosures for the letter.

InfoBlock Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

RecipientCode Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

RecipientGender Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

ReturnAddressShortForm Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

SenderCity Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

SenderCode Optional **Variant**. This argument may not be available to you,

depending on the language support (U.S. English, for example) that you've selected or installed.

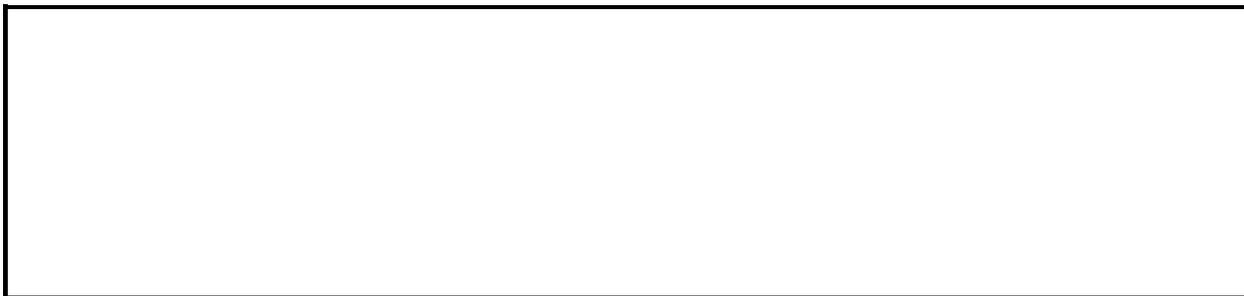
SenderGender Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

SenderReference Optional **Variant**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

Example

The following example uses the **CreateLetterContent** method to create a new **LetterContent** object in the active document and then uses this object with the **RunLetterWizard** method.

```
Set myLetter = ActiveDocument _  
    .CreateLetterContent(DateFormat:="July 31, 1996", _  
    IncludeHeaderFooter:=False, PageDesign:="", _  
    LetterStyle:=wdFullBlock, Letterhead:=True, _  
    LetterheadLocation:=wdLetterTop, _  
    LetterheadSize:=InchesToPoints(1.5), _  
    RecipientName:="Dave Edson", _  
    RecipientAddress:="436 SE Main St." & vbCr _  
    & "Bellevue, WA 98004", _  
    Salutation:="Dear Dave,", _  
    SalutationType:=wdSalutationInformal, _  
    RecipientReference:"", MailingInstructions:"", _  
    AttentionLine:"", Subject:="End of year report", _  
    CCList:"", ReturnAddress:"", _  
    SenderName:"", Closing:="Sincerely yours,", _  
    SenderCompany:"", SenderJobTitle:"", _  
    SenderInitials:"", EnclosureNumber:=0)  
ActiveDocument.RunLetterWizard LetterContent:=myLetter
```



↳ [Show All](#)

CreateNewDocument Method

▶ [CreateNewDocument method as it applies to the **MailingLabel** object.](#)

Creates a new label document using either the default label options or ones that you specify. Returns a **Document** object that represents the new document.

expression.**CreateNewDocument**(*Name*, *Address*, *AutoText*, *ExtractAddress*, *LaserTray*, *PrintEPostageLabel*, *Vertical*)

expression Required. An expression that returns one of the above objects.

Name Optional **Variant**. The mailing label name.

Address Optional **Variant**. The text for the mailing label.

AutoText Optional **Variant**. The name of the AutoText entry that includes the mailing label text.

ExtractAddress Optional **Variant**. **True** to use the address text marked by the user-defined bookmark named "EnvelopeAddress" instead of using the **Address** argument.

LaserTray Optional **Variant**. The laser printer tray. Can be one of the following **WdPaperTray** constants.

WdPaperTray can be any one of the following WdPaperTray constants:

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualFeed
wdPrinterManualEnvelopeFeed
wdPrinterMiddleBin
wdPrinterOnlyBin
wdPrinterPaperCassette
wdPrinterSmallFormatBin
wdPrinterTractorFeed
wdPrinterUpperBin

PrintEPostageLabel Optional **VARIANT**. **True** to print postage using an Internet e-postage vendor.

Vertical Optional **VARIANT**. **True** formats text vertically on the label. Used for Asian-language mailing labels.

▶ [CreateNewDocument method as it applies to the **Hyperlink** object.](#)

Creates a new document linked to the specified hyperlink.

expression. **CreateNewDocument**(*FileName*, *EditNow*, *Overwrite*)

expression Required. An expression that returns one of the above objects.

FileName Required **String**. The file name of the specified document.

EditNow Required **Boolean**. **True** to have the specified document open immediately in its associated editing environment. The default value is **True**.

Overwrite Required **Boolean**. **True** to overwrite any existing file of the same name in the same folder. **False** if any existing file of the same name is preserved and the **FileName** argument specifies a new file name. The default value is **False**.

Example

▶ [As it applies to the **MailingLabel** object.](#)

This example creates a new Avery 2160 minilabel document using a predefined address.

```
addr = "Dave Edson" & vbCr & "123 Skye St." _  
      & vbCr & "Our Town, WA 98004"  
Application.MailingLabel.CreateNewDocument _  
      Name:="2160 mini", Address:=addr, ExtractAddress:=False
```

This example creates a new Avery 5664 shipping-label document using the selected text as the address.

```
addr = Selection.Text  
Application.MailingLabel.CreateNewDocument _  
      Name:="5664", Address:=addr, _  
      LaserTray:=wdPrinterUpperBin
```

This example creates a new self-adhesive-label document using the EnvelopeAddress bookmark text as the address.

```
If ActiveDocument.Bookmarks.Exists("EnvelopeAddress") = True Then  
    Application.MailingLabel.CreateNewDocument _  
        Name:="Self Adhesive Tab 1 1/2""", ExtractAddress:=True  
End If
```

▶ [As it applies to the **Hyperlink** object.](#)

This example creates a new document based on the new hyperlink in the first document and then loads the new document into Microsoft Word for editing. The document is called "Overview.doc," and it overwrites any file of the same name in the \\Server1\Annual folder.

```
With Documents(1)  
    Set objHyper = _  
        .Hyperlinks.Add(Anchor:=Selection.Range, _  
        Address:="\\Server1\Annual\Overview.doc")  
    objHyper.CreateNewDocument _  
        FileName:="\\Server1\Annual\Overview.doc", _  
        EditNow:=True, Overwrite:=True
```

End With

--

CreateTextbox Method

-

Adds a default-size text box around the selection. If the selection is an insertion point, this method changes the pointer to a cross-hair pointer so that the user can draw a text box.

expression.**CreateTextbox**

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

Remarks

Using this method is equivalent to clicking the **Text Box** button on the **Drawing** toolbar. A text box is a rectangle with an associated text frame.

Example

This example adds a text box around the selection and then changes the text box's line style.

```
If Selection.Type = wdSelectionNormal Then
    Selection.CreateTextbox
    Selection.ShapeRange(1).Line.DashStyle =msoLineDashDot
End If
```



CustomDrop Method

-

Sets the vertical distance (in points) 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 (the place that the callout points to), in which case the drop distance is measured from the bottom of the text box.

expression.**CustomDrop**(*Drop*)

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

Drop Required **Single**. The drop distance, in points.

Remarks

If the [PresetDrop](#) method was previously used to set the drop for the specified callout, use the statement `PresetDrop msoCalloutDropCustom` before using the **CustomDrop** method so that the custom drop setting takes effect.

Example

This example cancels any preset drop that's been set for the first shape in the active document, 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 first shape must be a callout.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes(1).Callout
    .PresetDrop msoCalloutDropCustom
    .CustomDrop 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. 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**).

expression.**CustomLength**(*Length*)

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

Length Required **Single**. The length of the first segment of the callout, in points.

Remarks

Applying this method sets the [AutoLength](#) property to **False** and sets the [Length](#) property to the value specified for the **Length** argument.

Example

This example toggles between an automatically scaling first segment and one with a fixed length for the callout line for the first shape on the active document. For the example to work, the first shape must be a callout.

```
Dim docActive As Document
Set docActive = ActiveDocument
With docActive.Shapes(1).Callout
    If .AutoLength Then
        .CustomLength 50
    Else
        .AutomaticLength
    End If
End With
```



Cut Method

-
Removes the specified object from the document and places it on the Clipboard.

expression.Cut

expression Required. An expression that returns a **Field**, **FormField**, **Frame**, **MailMergeField**, **PageNumber**, **Range**, or **Selection** object.

Remarks

If *expression* returns a **Range** or **Selection** object, the contents of the object are cut to the Clipboard but the collapsed object remains in the document.

Example

This example cuts the first field in the active document and pastes the field at the insertion point.

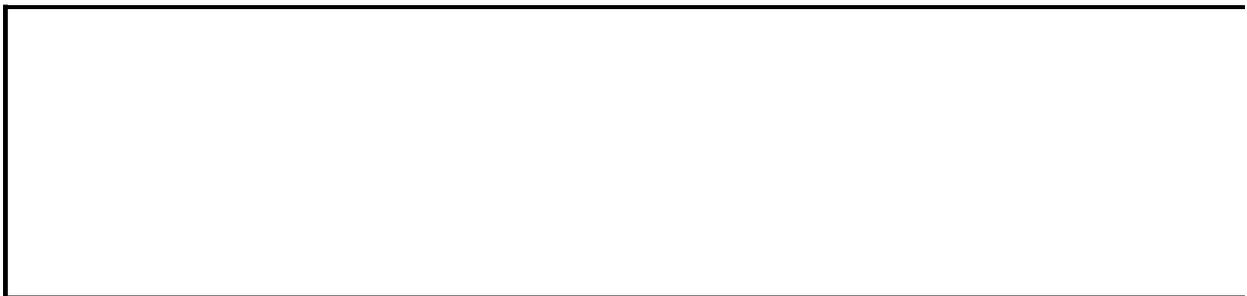
```
If ActiveDocument.Fields.Count >= 1 Then
    ActiveDocument.Fields(1).Cut
    Selection.Collapse Direction:=wdCollapseEnd
    Selection.Paste
End If
```

This example cuts the first word in the first paragraph and pastes the word at the end of the paragraph.

```
With ActiveDocument.Paragraphs(1).Range
    .Words(1).Cut
    .Collapse Direction:=wdCollapseEnd
    .Move Unit:=wdCharacter, Count:=-1
    .Paste
End With
```

This example cuts the contents of the selection and pastes them into a new document.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Cut
    Documents.Add.Content.Paste
End If
```



DataForm Method

-

Displays the **Data Form** dialog box, in which you can add, delete, or modify data records.

Note You can use this method with a mail merge main document, a mail merge data source, or any document that contains data delimited by table cells or separator characters.

expression.**DataForm**

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

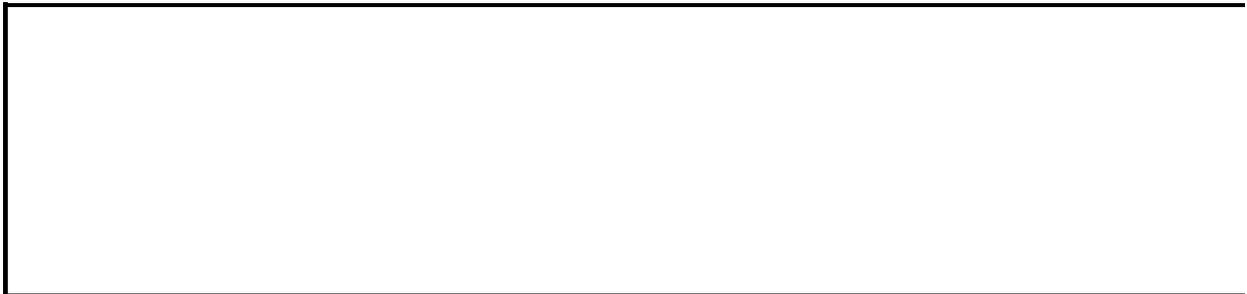
Example

This example displays the **Data Form** dialog box if the active document is a mail merge document.

```
If ActiveDocument.MailMerge.State <> wdNormalDocument Then
    ActiveDocument.DataForm
End If
```

This example creates a table in a new document and then displays the **Data Form** dialog box.

```
Set aDoc = Documents.Add
With aDoc
    .Tables.Add Range:=aDoc.Content, NumRows:=2, NumColumns:=2
    .Tables(1).Cell(1, 1).Range.Text = "Name"
    .Tables(1).Cell(1, 2).Range.Text = "Age"
    .DataForm
End With
```



DDEExecute Method

-
Sends a command or series of commands to an application through the specified dynamic data exchange (DDE) channel.

expression.**DDEExecute**(*Channel*, *Command*)

expression Optional. An expression that returns an **Application** object.

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Command Required **String**. A command or series of commands recognized by the receiving application (the DDE server). If the receiving application cannot perform the specified command, an error occurs.

Example

This example creates a new worksheet in Microsoft Excel. The XLM macro instruction to create a new worksheet is `New(1)`.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")  
DDEExecute Channel:=lngChannel, Command:="[New(1)]"  
DDETerminate Channel:=lngChannel
```

This example runs the Microsoft Excel macro named "Macro1" in Personal.xls.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")  
DDEExecute Channel:=lngChannel, Command:="[Run(" & Chr(34) & _  
    "Personal.xls!Macro1" & Chr(34) & ")]"  
DDETerminate Channel:=lngChannel
```



DDEInitiate Method

Opens a dynamic data exchange (DDE) channel to another application, and returns the channel number.

expression.**DDEInitiate**(*App*, *Topic*)

expression Optional. An expression that returns an **Application** object.

App Required **String**. The name of the application.

Topic Required **String**. The name of a DDE topic — for example, the name of an open document — recognized by the application to which you're opening a channel.

Remarks

If it's successful, the **DDEInitiate** method returns the number of the open channel. All subsequent DDE functions use this number to specify the channel.

Example

This example initiates a DDE conversation with the System topic and opens the Microsoft Excel workbook Sales.xls. The example terminates the DDE channel, initiates a channel to Sales.xls, and then inserts text into cell R1C1.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")  
DDEExecute Channel:=lngChannel, Command:="[OPEN(" & Chr(34) _  
    & "C:\Sales.xls" & Chr(34) & ")]"  
DDETerminate Channel:=lngChannel  
lngChannel = DDEInitiate(App:="Excel", Topic:="Sales.xls")  
DDEPoke Channel:=lngChannel, Item:="R1C1", Data:="1996 Sales"  
DDETerminate Channel:=lngChannel
```



DDEPoke Method

-
Uses an open dynamic data exchange (DDE) channel to send data to an application.

expression.DDEPoke(Channel, Item, Data)

expression Optional. An expression that returns an **Application** object.

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Item Required **String**. The item within a DDE topic to which the specified data is to be sent.

Data Required **String**. The data to be sent to the receiving application (the DDE server).

Remarks

If the **DDEPoke** method isn't successful, an error occurs.

Example

This example opens the Microsoft Excel workbook Sales.xls and inserts "1996 Sales" into cell R1C1.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")  
DDEExecute Channel:=lngChannel, Command:="[OPEN(" & Chr(34) _  
    & "C:\Sales.xls" & Chr(34) & ")"]  
DDETerminate Channel:=lngChannel  
lngChannel = DDEInitiate(App:="Excel", Topic:="Sales.xls")  
DDEPoke Channel:=lngChannel, Item:="R1C1", Data:="1996 Sales"  
DDETerminate Channel:=lngChannel
```



DDERequest Method

-
Uses an open dynamic data exchange (DDE) channel to request information from the receiving application, and returns the information as a string.

expression.**DDERequest**(*Channel*, *Item*)

expression Optional. An expression that returns an **Application** object.

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Item Required **String**. The item to be requested.

Remarks

When you request information from the topic in the server application, you must specify the item in that topic whose contents you're requesting. In Microsoft Excel, for example, cells are valid items, and you refer to them by using either the "R1C1" format or named references.

Microsoft Excel and other applications that support DDE recognize a topic named "System." Three standard items in the System topic are described in the following table. Note that you can get a list of the other items in the System topic by using the SysItems item.

Item in System topic	Effect
SysItems	Returns a list of all the items in the System topic.
Topics	Returns a list of all the available topics.
Formats	Returns a list of all the Clipboard formats supported by Word.

Example

This example opens the Microsoft Excel workbook Book1.xls and retrieves the contents of cell R1C1.

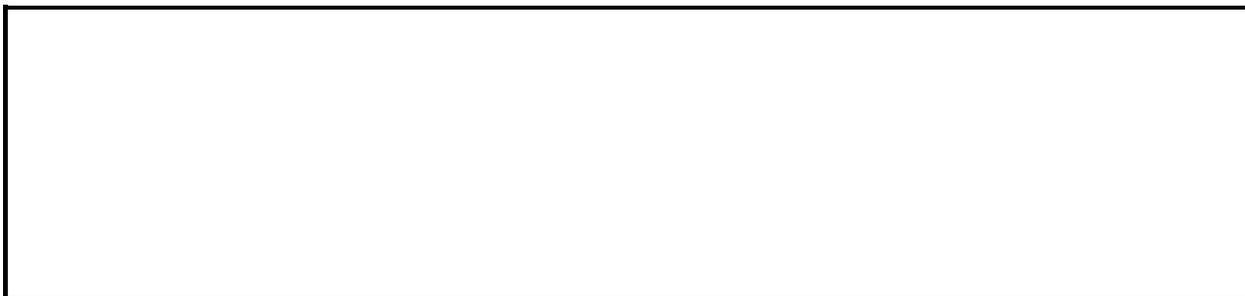
```
Dim lngChannel As Long

lngChannel = DDEInitiate(App:="Excel", Topic:="System")
DDEExecute Channel:=lngChannel, Command:="[OPEN(" & Chr(34) _
    & "C:\Documents\Book1.xls" & Chr(34) & ")]"
DDETerminate Channel:=lngChannel
lngChannel = DDEInitiate(App:="Excel", Topic:="Book1.xls")
MsgBox DDERequest(Channel:=lngChannel, Item:="R1C1")
DDETerminateAll
```

This example opens a channel to the System topic in Microsoft Excel and then uses the Topics item to return a list of available topics. The example inserts the topic list, which includes all open workbooks, after the selection.

```
Dim lngChannel As Long
Dim strTopicList As String

lngChannel = DDEInitiate(App:="Excel", Topic:="System")
strTopicList = DDERequest(Channel:=lngChannel, Item:="Topics")
Selection.InsertAfter strTopicList
DDETerminate Channel:=lngChannel
```



DDETerminate Method

-
Closes the specified dynamic data exchange (DDE) channel to another application.

expression.**DDETerminate**(*Channel*)

expression Optional. An expression that returns an **Application** object.

Channel Required **Long**. The channel number returned by the **DDEInitiate** method.

Example

This example creates a new workbook in Microsoft Excel and then terminates the DDE conversation.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")
```

```
DDEExecute Channel:=lngChannel, Command:="[New(1)]"
```

```
DDETerminate Channel:=lngChannel
```



DDETerminateAll Method

-

Closes all dynamic data exchange (DDE) channels opened by Word. This method doesn't close channels opened to Word by client applications. Using this method is the same as using the **DDETerminate** method for each open channel.

expression.**DDETerminateAll**

expression Optional. An expression that returns an **Application** object.

Remarks

If you interrupt a macro that opens a DDE channel, you may inadvertently leave a channel open. Open channels aren't closed automatically when a macro ends, and each open channel uses system resources. For this reason, it's a good idea to use this method when you're debugging a macro that opens one or more DDE channels.

Example

This example opens the Microsoft Excel workbook Book1.xls, inserts text into cell R2C3, saves the workbook. and then terminates all DDE channels.

```
Dim lngChannel As Long
```

```
lngChannel = DDEInitiate(App:="Excel", Topic:="System")  
DDEExecute Channel:=lngChannel, Command:="[OPEN(" & Chr(34) & _  
    "C:\Documents\Book1.xls" & Chr(34) & ")]"  
DDETerminate Channel:=lngChannel  
lngChannel = DDEInitiate(App:="Excel", Topic:="Book1.xls")  
DDEPoke Channel:=lngChannel, Item:="R2C3", Data:="Hello World"  
DDEExecute Channel:=lngChannel, Command:="[Save]"  
DDETerminateAll
```



DecreaseSpacing Method

-
Decreases the spacing before and after paragraphs in six-point increments.

expression.**DecreaseSpacing**

expression Required. An expression that returns a [Paragraphs](#) object.

Example

This example decreases the before and after spacing of a paragraph or selection of paragraphs by six points each time the procedure is run. If the before and after spacing are both zero, the procedure will do nothing.

```
Sub DecreaseParaSpacing()  
    Selection.Paragraphs.DecreaseSpacing  
End Sub
```



DefaultWebOptions Method

Returns the [DefaultWebOptions](#) object that contains global application-level attributes used by Microsoft Word whenever you save a document as a Web page or open a Web page.

expression.**DefaultWebOptions**

expression Required. An expression that returns an **Application** object.

Example

This example checks to see whether the default setting for document encoding is Western, and then it sets the string `strDocEncoding` accordingly.

```
Dim strDocEncoding As String

If Application.DefaultWebOptions.Encoding _
    = msoEncodingWestern Then
    strDocEncoding = "Western"
Else
    strDocEncoding = "Other"
End If
```



↳ [Show All](#)

Delete Method

▶ [Delete method as it applies to the **Cell** and **Cells** objects.](#)

Deletes a table cell or cells and optionally controls how the remaining cells are shifted.

expression.Delete(*ShiftCells*)

expression Required. An expression that returns one of the above objects.

ShiftCells Optional **Variant**. The direction in which the remaining cells are to be shifted. Can be any [WdDeleteCells](#) constant. If omitted, cells to the right of the last deleted cell are shifted left.

WdDeleteCells can be one of these WdDeleteCells constants.

wdDeleteCellsEntireColumn

wdDeleteCellsEntireRow

wdDeleteCellsShiftLeft

wdDeleteCellsShiftUp

▶ [Delete method as it applies to the **Range** and **Selection** objects.](#)

Deletes the specified number of characters or words. This method returns a **Long** value that indicates the number of items deleted, or it returns 0 (zero) if the deletion was unsuccessful.

expression.Delete(*Unit*, *Count*)

expression Required. An expression that returns one of the above objects.

Unit Optional **Variant**. The unit by which the collapsed range or selection is to be deleted. Can be one of the following **WdUnits** constants: **wdCharacter** (default) or **wdWord**.

Count Optional **Variant**. The number of units to be deleted. To delete units after the range or selection, collapse the range or selection and use a positive number. To delete units before the range or selection, collapse the range or selection and use a negative number.

▶ [Delete method as it applies to the **ShapeNodes** object.](#)

Deletes the specified object.

expression.**Delete**(*Index*)

expression Required. An expression that returns a [ShapeNodes](#) object.

Index Required **Long**. The number of the shape node to delete.

▶ [Delete method as it applies to all other objects in the Applies To list.](#)

Deletes the specified object.

expression.**Delete**

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

Example

▶ [As it applies to the **Cell** object.](#)

This example deletes the first cell in the first table of the active document.

```
Sub DeleteCells()  
    ActiveDocument.Tables(1).Cell(1, 1).Delete  
End Sub
```

▶ [As it applies to the **Range** and **Selection** objects.](#)

This example selects and deletes the contents of the active document.

```
Sub DeleteSelection()  
    ActiveDocument.Content.Select  
    Selection.Delete  
End Sub
```

This example collapses the selection and deletes the two words following the insertion point.

```
Sub DeleteSelection2()  
    ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(3).Range.S  
    Selection.Collapse Direction:=wdCollapseStart  
    Selection.Delete Unit:=wdWord, Count:=2  
End Sub
```

This example collapses myRange and deletes the two characters preceding the insertion point.

```
Sub DeleteRange()  
    Dim myRange As Range  
    Set myRange = Selection.Words(1)  
    myRange.Collapse Direction:=wdCollapseStart  
    myRange.Delete Unit:=wdCharacter, Count:=-2  
End Sub
```

This example deletes the first word in the active document.

```
Sub DeleteFirstWord()  
    ActiveDocument.Words(1).Delete
```

End Sub

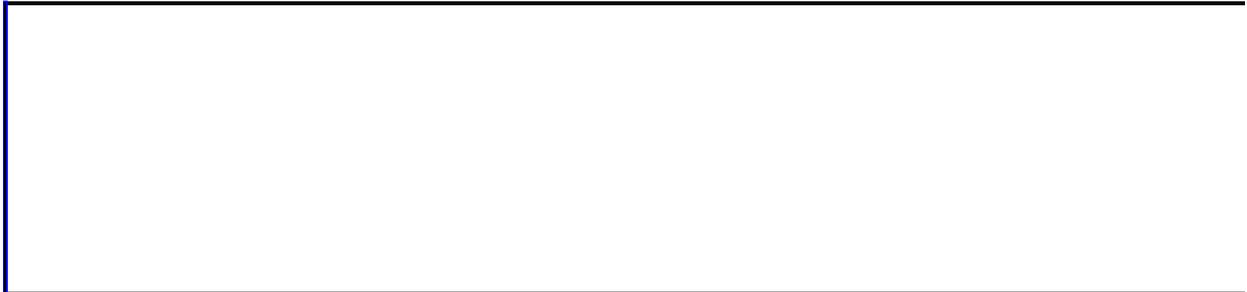
▶ [As it applies to other objects in the Applies To list.](#)

If a bookmark named "temp" exists in the active document, this example deletes the bookmark.

```
Sub DeleteBookmark()  
    If ActiveDocument.Bookmarks.Exists(Name:="temp") Then  
        ActiveDocument.Bookmarks(Name:="temp").Delete  
    End If  
End Sub
```

This example deletes the style named "Intro" from Sales.doc. Paragraphs using the Intro style will revert to using the Normal style.

```
Sub DeleteStyle()  
    Documents(Index:="Sales.doc").Styles _  
        (Index:="Intro").Delete  
End Sub
```



DeleteAllComments Method

-
Deletes all comments from the [Comments](#) collection in a document.

expression.**DeleteAllComments**

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

Remarks

Use the [Add](#) method for the **Comments** object to add a comment to a document.

Example

This example deletes all comments in the active document. This example assumes you have comments in active document.

```
Sub DelAllComments()  
    ActiveDocument.DeleteAllComments  
End Sub
```



DeleteAllCommentsShown Method

-
Deletes all revisions in a specified document that are displayed on the screen.

expression.DeleteAllCommentsShown

expression Required. An expression that returns a [Document](#) object.

Example

This example hides all comments made by "Jeff Smith" and deletes all other displayed comments.

```
Sub HideDeleteComments()  
    Dim rev As Reviewer  
    With ActiveWindow.View  
        'Display all comments and revisions  
        .ShowRevisionsAndComments = True  
        .ShowFormatChanges = True  
        .ShowInsertionsAndDeletions = True  
  
        For Each rev In .Reviewers  
            rev.Visible = True  
        Next  
  
        'Hide only the revisions/comments made by the  
        'reviewer named "Jeff Smith"  
        .Reviewers(Index:="Jeff Smith").Visible = False  
    End With  
  
    'Delete all comments displayed in the active view  
    ActiveDocument.DeleteAllCommentsShown  
End Sub
```



DetectLanguage Method

-
Analyzes the specified text to determine the language that it is written in.

expression.**DetectLanguage**

expression Required. An expression that returns a **Document**, **Range**, or **Selection** object.

Remarks

The results of the **DetectLanguage** method are stored in the [LanguageID](#) property on a character-by-character basis. To read the **LanguageID** property, you must first specify a selection or range of text.

When applied to a **Document** object, the **DetectLanguage** method checks all available text in the document (headers, footers, text boxes, and so forth). If the specified text contains a partial sentence, the selection or range is extended to the end of the sentence.

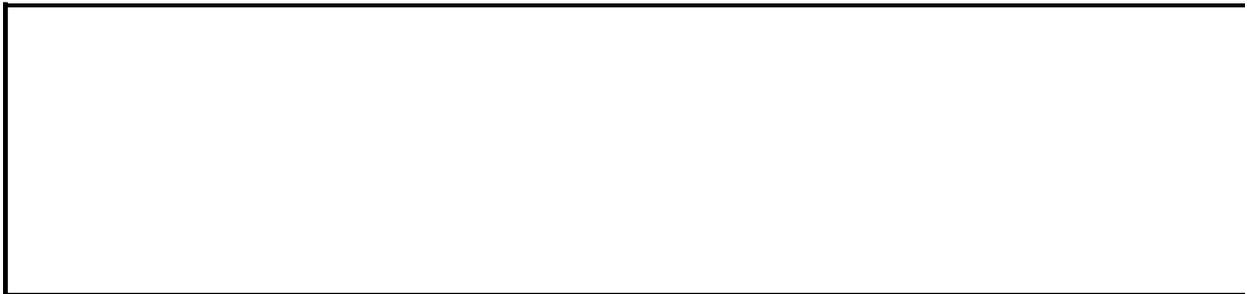
If the **DetectLanguage** method has already been applied to the specified text, the [LanguageDetected](#) property is set to **True**. To reevaluate the language of the specified text, you must first set the **LanguageDetected** property to **False**.

For more information about automatic language detection, see [About automatic language detection](#).

Example

This example checks the active document to determine the language it's written in and then displays the result.

```
With ActiveDocument
  If .LanguageDetected = True Then
    x = MsgBox("This document has already " _
      & "been checked. Do you want to check " _
      & "it again?", vbYesNo)
    If x = vbYes Then
      .LanguageDetected = False
      .DetectLanguage
    End If
  Else
    .DetectLanguage
  End If
  If .Range.LanguageID = wdEnglishUS Then
    MsgBox "This is a U.S. English document."
  Else
    MsgBox "This is not a U.S. English document."
  End If
End With
```



Disable Method

-

Removes the specified key combination if it's currently assigned to a command. After you use this method, the key combination has no effect. Using this method is the equivalent to clicking the **Remove** button in the **Customize Keyboard** dialog box (**Tools** menu).

Note Use the [Clear](#) method with a **KeyBinding** object to reset a built-in command to its default key assignment. You don't need to remove or rebind a **KeyBinding** object before adding it elsewhere.

expression.**Disable**

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

Example

This example removes the CTRL+SHIFT+B key assignment. This key combination is assigned to the **Bold** command by default.

```
CustomizationContext = NormalTemplate
FindKey(BuildKeyCode(wdKeyControl, wdKeyShift, wdKeyB)).Disable
```

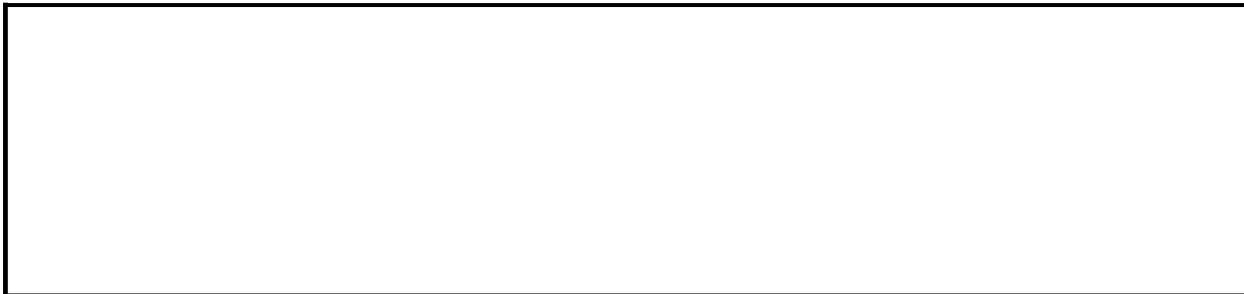
This example assigns the CTRL+SHIFT+O key combination to the **Organizer** command. The example then uses the **Disable** method to remove the CTRL+SHIFT+O key combination and displays a message.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyO, _
    wdKeyControl, wdKeyShift), _
    KeyCategory:=wdKeyCategoryCommand, Command:="Organizer"
With FindKey(BuildKeyCode(wdKeyO, wdKeyControl, wdKeyShift))
    MsgBox .Command & " is assigned to CTRL+Shift+O"
    .Disable
    If .Command = "" Then MsgBox _
        "Nothing is assigned to CTRL+Shift+O"
End With
```

This example removes all key assignments for the global macro named "Macro1."

```
Dim kbLoop As KeyBinding

CustomizationContext = NormalTemplate
For Each kbLoop In KeysBoundTo _
    (KeyCategory:=wdKeyCategoryMacro, Command:="Macro1")
    kbLoop.Disable
Next kbLoop
```



This keyword is not implemented. It is reserved for future use.

Display Method

-

Displays the specified built-in Word dialog box until either the user closes it or the specified amount of time has passed. Returns a **Long** that indicates which button was clicked to close the dialog box.

Return value	Description
-2	The Close button.
-1	The OK button.
0 (zero)	The Cancel button.
> 0 (zero)	A command button: 1 is the first button, 2 is the second button, and so on.

Note Any actions initiated or settings specified while a dialog box is displayed using this method aren't carried out. Use the [Show](#) method to display a dialog box and carry out actions or apply settings.

expression.**Display**(*TimeOut*)

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

TimeOut Optional **Variant**. The amount of time that Word will wait before closing the dialog box automatically. One unit is approximately 0.001 second. Concurrent system activity may increase the effective time value. If this argument is omitted, the dialog box is closed when the user closes it.

Example

This example displays the **About** dialog box.

```
Dim dlgAbout As Dialog
```

```
Set dlgAbout = Dialogs(wdDialogHelpAbout)  
dlgAbout.Display
```

This example displays the **Zoom** dialog box for approximately nine seconds.

```
Dialogs(wdDialogViewZoom).Display TimeOut:=9000
```



DisplayMoveDialog Method

-

Displays the **Move** dialog box, in which the user can specify a new location for the active e-mail message in an available message store. This method is available only if you are using Word as your e-mail editor.

expression.**DisplayMoveDialog**

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

Example

This example displays the **Move** dialog box for the active e-mail message.

`Application.MailMessage.DisplayMoveDialog`



DisplayProperties Method

-

Displays the **Properties** dialog box for the active e-mail message. This method is available only if you are using Word as your e-mail editor.

expression.**DisplayProperties**

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

Example

This example displays the **Properties** dialog box for the active e-mail message.

```
Application.MailMessage.DisplayProperties
```



DisplaySelectNamesDialog Method

-

Displays the **Select Names** dialog box, in which the user can add addresses to the **To:**, **Cc:**, and **Bcc:** lines in the active, unsent e-mail message. This method is available only if you are using Word as your e-mail editor.

expression.**DisplaySelectNamesDialog**

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

Example

This example displays the **Select Names** dialog box for the active e-mail message.

```
Application.MailMessage.DisplaySelectNamesDialog
```



Distribute Method

-

Evenly distributes the shapes in the specified range of shapes. You can specify whether you want to distribute the shapes horizontally or vertically and whether you want to distribute them over the entire page or just over the space they originally occupy.

expression.**Distribute**(*Distribute*, *RelativeTo*)

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

Distribute Required [MsoDistributeCmd](#).

MsoDistributeCmd can be one of these MsoDistributeCmd constants.

msoDistributeHorizontally

msoDistributeVertically

RelativeTo Required **Long**. **True** to distribute the shapes evenly over the entire horizontal or vertical space on the page. **False** to distribute them within the horizontal or vertical space that the range of shapes originally occupies.

Example

This example defines a shape range that contains all the AutoShapes on the active document and then horizontally distributes the shapes in this range.

```
With ActiveDocument.Shapes
    numShapes = .Count
    If numShapes > 1 Then
        numAutoShapes = 0
        ReDim autoShpArray(1 To numShapes)
        For i = 1 To numShapes
            If .Item(i).Type = msoAutoShape Then
                numAutoShapes = numAutoShapes + 1
                autoShpArray(numAutoShapes) = .Item(i).Name
            End If
        Next
        If numAutoShapes > 1 Then
            ReDim Preserve autoShpArray(1 To numAutoShapes)
            Set asRange = .Range(autoShpArray)
            asRange.Distribute msoDistributeHorizontally, False
        End If
    End If
End With
```



DistributeHeight Method

-
Adjusts the height of the specified rows or cells so that they're equal.

expression.**DistributeHeight**

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

Example

This example adjusts the height of the rows in the first table in the active document so that they're equal.

```
ActiveDocument.Tables(1).Rows.DistributeHeight
```

This example adjusts the height of the first three rows in the first table so that they're equal.

```
Dim rngTemp As Range
```

```
If ActiveDocument.Tables.Count >= 1 Then  
    Set rngTemp = ActiveDocument.Range(Start:=ActiveDocument _  
        .Tables(1).Rows(1).Range.Start, _  
        End:=ActiveDocument.Tables(1).Rows(3).Range.End)  
    rngTemp.Rows.DistributeHeight  
End If
```



DistributeWidth Method

-
Adjusts the width of the specified columns or cells so that they're equal.

expression.**DistributeWidth**

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

Example

This example adjusts the width of the columns in the first table in the active document so that they're equal.

```
ActiveDocument.Tables(1).Columns.DistributeWidth
```

This example adjusts the height of the selected cells.

```
If Selection.Cells.Count >= 2 Then  
    Selection.Cells.DistributeWidth  
End If
```



DoClick Method

-

Clicks the specified field. If the field is a GOTOBUTTON field, this method moves the insertion point to the specified location or selects the specified bookmark. If the field is a MACROBUTTON field, this method runs the specified macro. If the field is a HYPERLINK field, this method jumps to the target location.

expression.**DoClick**

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

Example

If the first field in the selection is a GOTOBUTTON field, this example clicks it (the insertion point is moved to the specified location, or the specified bookmark is selected).

```
Dim fldTemp
```

```
Set fldTemp = Selection.Fields(1)
```

```
If fldTemp.Type = wdFieldGoToButton Then fldTemp.DoClick
```



DoVerb Method

Requests that an OLE object perform one of its available verbs — the actions an OLE object takes to activate its contents. Each OLE object supports a set of verbs that pertain to that object.

expression.**DoVerb**(*VerbIndex*)

expression Required. An expression that returns an **OLEFormat** object.

VerbIndex Optional **Variant**. The verb that the OLE object should perform. If this argument is omitted, the default verb is sent. If the OLE object does not support the requested verb, an error will occur. Can be any [WdOLEVerb](#) constant.

WdOLEVerb can be one of these WdOLEVerb constants.

wdOLEVerbPrimary Performs the verb that is invoked when the user double-clicks the object.

wdOLEVerbShow Shows the object to the user for editing or viewing. Use it to show a newly inserted object for initial editing.

wdOLEVerbOpen Opens the object in a separate window.

wdOLEVerbHide Removes the object's user interface from view.

wdOLEVerbUIActivate Activates the object in place and displays any user-interface tools that the object needs, such as menus or toolbars.

wdOLEVerbInPlaceActivateRuns the object and installs its window, but doesn't install any user-interface tools.

wdOLEVerbDiscardUndoState Forces the object to discard any undo state that it might be maintaining; note that the object remains active, however.

Example

This example sends the default verb to the server for the first floating OLE object on the active document.

```
ActiveDocument.Shapes(1).OLEFormat.DoVerb
```



Duplicate Method

-

Creates a duplicate of the specified **Shape** or **ShapeRange** object, adds the new range of shapes to the **Shapes** collection at a standard offset from the original shapes, and then returns the new **Shape** object.

expression.**Duplicate**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Example

This example creates a duplicate of shape one on the active document and then changes the fill for the new shape.

```
Set newShape = ActiveDocument.Shapes(1).Duplicate  
With newShape  
    .Fill.PresetGradient msoGradientVertical, 1, msoGradientGold  
End With
```



Edit Method

-
Opens the specified OLE object for editing in the application it was created in.

expression.**Edit**

expression Required. An expression that returns an **OLEFormat** object.

Example

This example opens (for editing) the first embedded OLE object (defined as a shape) on the active document.

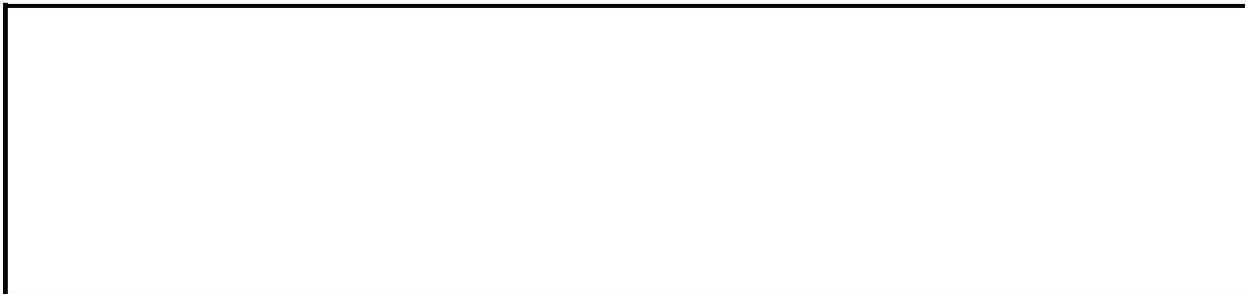
```
Dim shapesAll As Shapes

Set shapesAll = ActiveDocument.Shapes
If shapesAll.Count >= 1 Then
    If shapesAll(1).Type = msoEmbeddedOLEObject Then
        shapesAll(1).OLEFormat.Edit
    End If
End If
```

This example opens (for editing) the first linked OLE object (defined as an inline shape) in the active document.

```
Dim colIS As InlineShapes

Set colIS = ActiveDocument.InlineShapes
If colIS.Count >= 1 Then
    If colIS(1).Type = wdInlineShapeLinkedOLEObject Then
        colIS(1).OLEFormat.Edit
    End If
End If
```



EditDataSource Method

-
Opens or switches to the mail merge data source.

expression.**EditDataSource**

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

Remarks

If the data source is a Word document, this method opens the data source (or activates the data source if it's already open).

If Word is accessing the data through dynamic data exchange (DDE) — using an application such as Microsoft Excel or Microsoft Access — this method displays the data source in that application.

If Word is accessing the data through open database connectivity (ODBC), this method displays the data in a Word document. Note that if Microsoft Query is installed, a message appears, providing the option to display Microsoft Query instead of converting data.

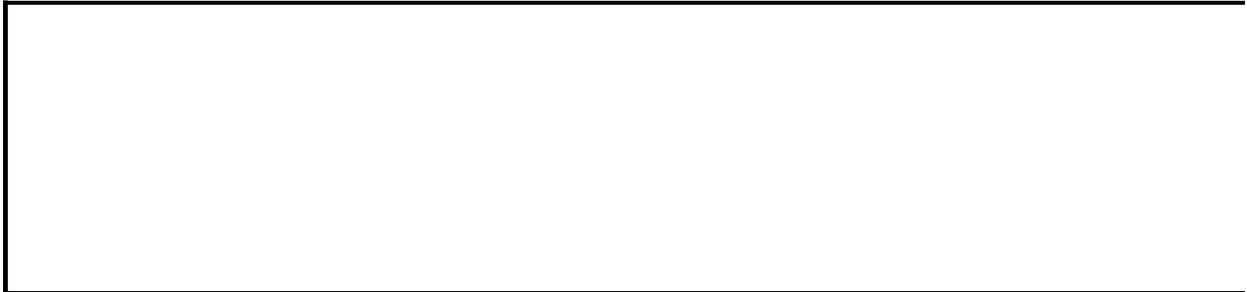
Example

This example opens or activates the data source attached to the document named "Sales.doc."

```
Documents("Sales.doc").MailMerge.EditDataSource
```

This example opens or activates the attached data source if the data source is a Word document.

```
Dim dsMain As MailMergeDataSource  
  
Set dsMain = ActiveDocument.MailMerge.DataSource  
If dsMain.Type = wdMergeInfoFromWord Then  
    ActiveDocument.MailMerge.EditDataSource  
End If
```



EditHeaderSource Method

-

Opens the header source attached to a mail merge main document, or activates the header source if it's already open.

Note If the mail merge main document doesn't have a header source, this method causes an error.

expression.**EditHeaderSource**

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

Example

This example attaches a header source to the active document and then opens the header source.

```
With ActiveDocument.MailMerge
    .MainDocumentType = wdFormLetters
    .OpenHeaderSource Name:="C:\Documents\Header.doc"
    .EditHeaderSource
End With
```

This example opens the header source if the active document has an associated header file attached to it.

```
Dim mmTemp As MailMerge

Set mmTemp = ActiveDocument.MailMerge
If mmTemp.State = wdMainAndSourceAndHeader Or _
    mmTemp.State = wdMainAndHeader Then
    mmTemp.EditHeaderSource
End If
```



EditMainDocument Method

-
Activates the mail merge main document associated with the specified header source or data source document.

Note If the main document isn't open, an error occurs. Use the [Open](#) method if the main document isn't currently open.

expression.**EditMainDocument**

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

Example

This example attempts to activate the main document associated with the active data source document. If the main document isn't open, the **Open** dialog box is displayed, with a message in the status bar.

```
Sub ActivateMain()  
    On Error GoTo errorHandler  
    Documents("Data.doc").MailMerge.EditMainDocument  
  
    Exit Sub  
  
errorHandler:  
    If Err = 4605 Then StatusBar = "Main document is not open"  
    Dialogs(wdDialogFileOpen).Show  
End Sub
```



EditType Method

Sets options for the specified text form field.

expression.**EditType**(*Type*, *Default*, *Format*, *Enabled*)

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

Type Required [WdTextFormFieldType](#). The text box type.

WdTextFormFieldType can be one of these WdTextFormFieldType constants.

wdCalculationText

wdCurrentDateText

wdCurrentTimeText

wdDateText

wdNumberText

wdRegularText

Default Optional **Variant**. The default text that appears in the text box.

Format Optional **Variant**. The formatting string used to format the text, number, or date (for example, "0.00," "Title Case," or "M/d/yy"). For more examples of formats, see the list of formats for the specified text form field type in the **Text Form Field Options** dialog box.

Enabled Optional **Variant**. **True** to enable the form field for text entry.

Example

This example adds a text form field named "Today" at the beginning of the active document. The **EditType** method is used to set the type to **wdCurrentDateText** and set the date format to "M/d/yy."

```
With ActiveDocument.FormFields.Add _  
    (Range:=ActiveDocument.Range(0, 0), _  
     Type:=wdFieldFormTextInput)  
    .Name = "Today"  
    .TextInput.EditType Type:=wdCurrentDateText, _  
     Format:="M/d/yy", Enabled:=False  
End With
```



Enable Method

-
Formats the first character in the specified paragraph as a dropped capital letter.

expression.**Enable**

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

Example

This example formats the first paragraph in the selection to begin with a dropped capital letter.

```
With Selection.Paragraphs(1).DropCap
    .Enable
    .LinesToDrop = 2
    .FontName = "Arial"
End With
```



EndKey Method

Moves or extends the selection to the end of the specified unit. This method returns an integer that indicates the number of characters the selection or active end was actually moved, or it returns 0 (zero) if the move was unsuccessful.

Note This method corresponds to functionality of the END key.

expression.**EndKey**(*Unit*, *Extend*)

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

Unit Optional **Variant**. The unit by which the selection is to be moved or extended. [WdUnits](#).

Can be one of the following **WdUnits** constants:

wdStory

wdColumn

wdLine

wdRow. The default value is **wdLine**.

Extend Optional **Variant**. Specifies the way the selection is moved. [WdMovementType](#).

Can be one of the following **WdMovementType** constants:

wdMove

wdExtend.

If the value of this argument is **wdMove**, the selection is collapsed to an insertion point and moved to the end of the specified unit. If it's **wdExtend**, the

end of the selection is extended to the end of the specified unit. The default value is **wdMove**.

Example

This example moves the selection to the end of the current line and assigns the number of characters moved to the pos variable.

```
pos = Selection.EndKey(Unit:=wdLine, Extend:=wdMove)
```

This example moves the selection to the beginning of the current table column and then extends the selection to the end of the column.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.HomeKey Unit:=wdColumn, Extend:=wdMove  
    Selection.EndKey Unit:=wdColumn, Extend:=wdExtend  
End If
```

This example moves the selection to the end of the current story. If the selection is in the main text story, the example moves the selection to the end of the document.

```
Selection.EndKey Unit:=wdStory, Extend:=wdMove
```



EndOf Method

Moves or extends the ending character position of a range or selection to the end of the nearest specified text unit. This method returns a value that indicates the number of character positions the range or selection was moved or extended (movement is forward in the document).

expression.**EndOf**(*Unit*, *Extend*)

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

Unit Optional **Variant**. The unit by which to move the ending character position. [WdUnits](#).

Can be one of the following **WdUnits** constants:

wdCharacter

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable.

If *expression* returns a **Selection** object, **wdLine** can also be used. The default value is **wdWord**.

Extend Optional Variant. [WdMovementType](#).

Can be either of the following **WdMovementType** constants:

wdMove

wdExtend

If **wdMove**, both ends of the range or selection object are moved to the end of the specified unit. If **wdExtend** is used, the end of the range or selection is extended to the end of the specified unit. The default value is **wdMove**.

Remarks

If the both the starting and ending positions for the range or selection are already at the end of the specified unit, this method doesn't move or extend the range or selection. For example, if the selection is at the end of a word and the trailing space, the following instruction doesn't change the selection (char equals 0 (zero)).

```
char = Selection.EndOf(Unit:=wdWord, Extend:=wdMove)
```

Example

This example extends the selection to the end of the paragraph.

```
charmoved = Selection.EndOf(Unit:=wdParagraph, Extend:=wdExtend)
If charmoved = 0 Then MsgBox "Selection unchanged"
```

This example moves myRange to the end of the first word in the selection (after the trailing space).

```
Set myRange = Selection.Characters(1)
myRange.EndOf Unit:=wdWord, Extend:=wdMove
```

This example adds a table, selects the first cell in row two, and then extends the selection to the end of the column.

```
Set myRange = ActiveDocument.Range(0, 0)
Set myTable = ActiveDocument.Tables.Add(Range:=myRange, _
    NumRows:=5, NumColumns:=3)
myTable.Cell(2, 1).Select
Selection.EndOf Unit:=wdColumn, Extend:=wdExtend
```



EndReview Method

-
Terminates a review of a file that has been sent for review using the [SendForReview](#) method or that has been automatically placed in a review cycle by sending a document to another user in an e-mail message.

expression.**EndReview**

expression Required. An expression that returns a [Document](#) object.

Remarks

When executed, the **EndReview** method displays a message asking the user whether to end the review.

Example

This example terminates the review of the active document. This example assumes the active document part of a review cycle.

```
Sub EndDocRev()  
    ActiveDocument.EndReview  
End Sub
```



EscapeKey Method

-
Cancels a mode such as extend or column select (equivalent to pressing the ESC key).

expression.**EscapeKey**

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

Example

This example turns on and then cancels extend mode.

```
With Selection  
    .ExtendMode = True  
    .EscapeKey  
End With
```



↳ [Show All](#)

Execute Method

▶ [Execute method as it applies to the **Find** object.](#)

Runs the specified find operation. Returns **True** if the find operation is successful. **Boolean**.

expression.Execute(FindText, MatchCase, MatchWholeWord, MatchWildcards, MatchSoundsLike, MatchAllWordForms, Forward, Wrap, Format, ReplaceWith, Replace, MatchKashida, MatchDiacritics, MatchAlefHamza, MatchControl)

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

FindText Optional **Variant**. The text to be searched for. Use an empty string ("") to search for formatting only. 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. For a list of special characters you can use, see [Find and replace text or other items](#).

MatchCase Optional **Variant**. **True** to specify that the find text be case sensitive. Corresponds to the **Match case** check box in the **Find and Replace** dialog box (**Edit** menu).

MatchWholeWord Optional **Variant**. **True** to have the find operation locate only entire words, not text that's part of a larger word. Corresponds to the **Find whole words only** check box in the **Find and Replace** dialog box.

MatchWildcards Optional **Variant**. **True** to have the find text be a special search operator. Corresponds to the **Use wildcards** check box in the **Find and Replace** dialog box.

MatchSoundsLike Optional **Variant**. **True** to have the find operation locate words that sound similar to the find text. Corresponds to the **Sounds like** check box in the **Find and Replace** dialog box.

MatchAllWordForms Optional **Variant**. **True** to have the find operation locate all forms of the find text (for example, "sit" locates "sitting" and "sat"). Corresponds to the **Find all word forms** check box in the **Find and Replace** dialog box.

Forward Optional **Variant**. **True** to search forward (toward the end of the document).

Wrap Optional **Variant**. Controls what happens if the search begins at a point other than the beginning of the document and the end of the document is reached (or vice versa if **Forward** is set to **False**). This argument also controls what happens if there's a selection or range and the search text isn't found in the selection or range. Can be one of the following [WdFindWrap](#) constants.

WdFindWrap can be one of these WdFindWrap constants.

wdFindAsk After searching the selection or range, Microsoft Word displays a message asking whether to search the remainder of the document.

wdFindContinue The find operation continues if the beginning or end of the search range is reached.

wdFindStop The find operation ends if the beginning or end of the search range is reached.

Format Optional **Variant**. **True** to have the find operation locate formatting in addition to or instead of the find text.

ReplaceWith Optional **Variant**. The replacement text. To delete the text specified by the **Find** argument, use an empty string (""). You specify special characters and advanced search criteria just as you do for the **Find** argument. To specify a graphic object or other nontext item as the replacement, move the item to the Clipboard and specify "^c" for **ReplaceWith**.

Replace Optional **Variant**. Specifies how many replacements are to be made: one, all, or none. Can be any [WdReplace](#) constant.

WdReplace can be one of these WdReplace constants.

wdReplaceAll

wdReplaceNone

wdReplaceOne

MatchKashida Optional **Variant. True** if find operations match text with matching kashidas in an Arabic language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

MatchDiacritics Optional **Variant. True** if find operations match text with matching diacritics in a right-to-left language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

MatchAlefHamza Optional **Variant. True** if find operations match text with matching Alef Hamzas in an Arabic language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

MatchControl Optional **Variant. True** if find operations match text with matching bidirectional control characters in a right-to-left language document. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

Remarks

If **MatchWildcards** is **True**, you can specify wildcard characters and other advanced search criteria for the **FindText** argument. For example, **"*(ing)"** finds any word that ends in "ing."

To search for a symbol character, type a caret (^), a zero (0), and then the symbol's character code. For example, **"^0151"** corresponds to an em dash (—).

Unless otherwise specified, replacement text inherits the formatting of the text it replaces in the document. For example, if you replace the string "abc" with "xyz," occurrences of "abc" with bold formatting are replaced with the string "xyz" with bold formatting.

Also, if **MatchCase** is **False**, occurrences of the search text that are uppercase will be replaced with an uppercase version of the replacement text regardless of the case of the search and replacement text. Using the previous example, occurrences of "ABC" are replaced with "XYZ."

▶ [Execute method as it applies to the **Dialog** and **KeyBinding** objects.](#)

Dialog object: Applies the current settings of a Microsoft Word dialog box.

KeyBinding object: Runs the command associated with the specified key combination.

expression.**Execute**

expression Required. An expression that returns one of the above objects.

▶ [Execute method as it applies to the **MailMerge** object.](#)

Performs the specified mail merge operation.

expression.**Execute(Pause)**

expression Required. An expression that returns one of the above objects.

Pause Optional **Variant**. **True** for Microsoft Word pause and display a

troubleshooting dialog box if a mail merge error is found. **False** to report errors in a new document.

Example

▶ [As it applies to the **Find** object.](#)

This example finds and selects the next occurrence of the word "library."

```
With Selection.Find
    .ClearFormatting
    .MatchWholeWord = True
    .MatchCase = False
    .Execute FindText:="library"
End With
```

This example finds all occurrences of the word "hi" in the active document and replaces each occurrence with "hello."

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="hi", _
    ReplaceWith:="hello", Replace:=wdReplaceAll
```

▶ [As it applies to the **Dialog** object.](#)

The following example enables the **Keep with next** check box on the **Line and Page Breaks** tab in the **Paragraph** dialog box.

```
With Dialogs(wdDialogFormatParagraph)
    .KeepWithNext = 1
    .Execute
End With
```

▶ [As it applies to the **KeyBinding** object.](#)

This example assigns the CTRL+SHIFT+C key combination to the **FileClose** command and then executes the key combination (the document is closed).

```
CustomizationContext = ActiveDocument.AttachedTemplate
Keybindings.Add KeyCode:=BuildKeyCode(wdKeyControl, _
    wdKeyShift, wdKeyC), KeyCategory:=wdKeyCategoryCommand, _
    Command:="FileClose"
FindKey(BuildKeyCode(wdKeyControl, wdKeyShift, wdKeyC)).Execute
```

▶ [As it applies to the **MailMerge** object.](#)

This example executes a mail merge if the active document is a main document with an attached data source.

```
Set myMerge = ActiveDocument.MailMerge  
If myMerge.State = wdMainAndDataSource Then MyMerge.Execute
```



Exists Method

-
Determines whether the specified bookmark or task exists. Returns **True** if the bookmark or task exists.

expression.**Exists**(*Name*)

expression An expression that returns a **Bookmarks** or **Tasks** object.

Name Required **String**. A bookmark or task name.

Example

This example determines whether the bookmark named "start" exists in the active document. If the bookmark exists, it's deleted.

```
If ActiveDocument.Bookmarks.Exists("start") = True Then
    ActiveDocument.Bookmarks("start").Delete
End If
```

This example determines whether the Windows Calculator program is running (if the task exists). If Calculator isn't running, the **Shell** statement starts it. If Calculator is running, the application is activated.

```
If Tasks.Exists("Calculator") = False Then
    Shell "Calc.exe"
Else
    Tasks("Calculator").Activate
End If
Tasks("Calculator").WindowState = wdWindowStateNormal
```



ExitWindows Method

-

Closes all open applications, quits Microsoft Windows, and logs the current user off. This method doesn't save changes to open Word documents; however, it does prompt you to save changes to open documents in other Windows-based applications.

expression.**ExitWindows**

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

Example

This example saves all open Word documents, quits Word, and then quits Microsoft Windows.

```
Documents.Save NoPrompt:=True, _  
    OriginalFormat:=wdOriginalDocumentFormat  
Tasks.ExitWindows
```



Expand Method

-

Expands the specified range or selection. Returns the number of characters added to the range or selection. **Long**

expression.**Expand**(*Unit*)

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

Unit Optional **Variant**. The unit by which to expand the range. [WdUnits](#).

Can be one of the following **WdUnits** constants:

wdCharacter

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable.

If *expression* represents a **Selection** object, **wdLine** can also be used. The default value is **wdWord**.

Example

This example creates a range that refers to the first word in the active document, and then it expands the range to reference the first paragraph in the document.

```
Set myRange = ActiveDocument.Words(1)
myRange.Expand Unit:=wdParagraph
```

This example capitalizes the first character in the selection and then expands the selection to include the entire sentence.

```
With Selection
    .Characters(1).Case = wdTitleSentence
    .Expand Unit:=wdSentence
End With
```



ExpandOutline Method

-
Expands the text under the selection or the specified range by one heading level.

Note If the document isn't in outline or master document view, an error occurs.

expression.**ExpandOutline**(*Range*)

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

Range Optional **Range** object. The range of paragraphs to be expanded. If this argument is omitted, the entire selection is expanded.

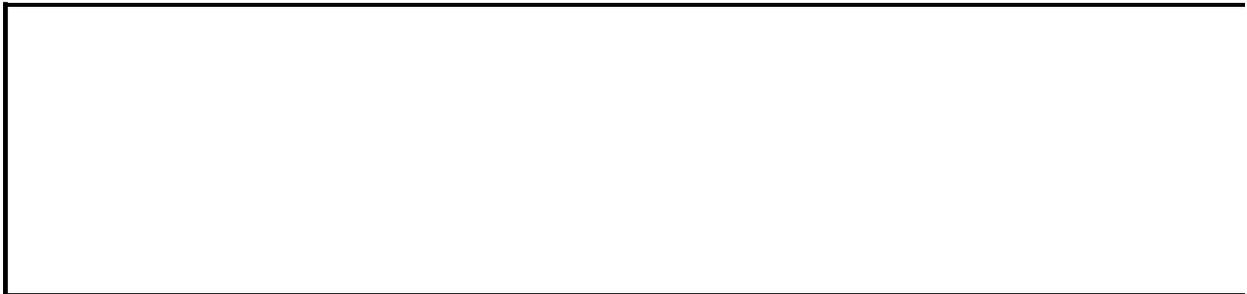
Example

This example expands every heading in the document by one level.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdOutlineView
    .ExpandOutline Range:=ActiveDocument.Content
End With
```

This example expands the active paragraph in the Document2 window.

```
With Windows("Document2")
    .Activate
    .View.Type = wdOutlineView
    .View.ExpandOutline
End With
```



Extend Method

-

Turns extend mode on (sets the [ExtendMode](#) property to **True**), or if extend mode is already on, extends the selection to the next larger unit of text. The progression of selected units of text is as follows: word, sentence, paragraph, section, entire document.

If **Character** is specified, extends the selection forward through the next instance of the specified character. The selection is extended by moving the active end of the selection.

expression.**Extend**(**Character**)

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

Character Optional **Variant**. The character through which the selection is extended. This argument is case sensitive and must evaluate to a **String** or an error occurs. Also, if the value of this argument is longer than a single character, Microsoft Word ignores the command entirely.

Example

This example collapses the current selection to an insertion point and then selects the current sentence.

```
With Selection
  ' Collapse current selection to insertion point.
  .Collapse
  ' Turn extend mode on.
  .Extend
  ' Extend selection to word.
  .Extend
  ' Extend selection to sentence.
  .Extend
End With
```

Here is an example that accomplishes the same task without the **Extend** method.

```
With Selection
  ' Collapse current selection.
  .Collapse
  ' Expand selection to current sentence.
  .Expand Unit:=wdSentence
End With
```

This example makes the end of the selection active and extends the selection through the next instance of a capital "R".

```
With Selection
  .StartIsActive = False
  .Extend Character:="R"
End Wit
```



FindRecord Method

-

Searches the contents of the specified mail merge data source for text in a particular field. Returns **True** if the search text is found. **Boolean**.

Note Corresponds to the **Find Record** button on the **Mail Merge** toolbar.

expression.**FindRecord**(*FindText*, *Field*)

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

FindText Required **String**. The text to be looked for.

Field Required **Variant**. The name of the field to be searched.

Example

This example displays a merge document for the first data record in which the FirstName field contains "Joe." If the data record is found, the number of the record is stored in the numRecord variable.

```
Dim dsMain As MailMergeDataSource
Dim numRecord As Integer

ActiveDocument.MailMerge.ViewMailMergeFieldCodes = False
Set dsMain = ActiveDocument.MailMerge.DataSource
If dsMain.FindRecord(FindText:="Joe", _
    Field:="FirstName") = True Then
    numRecord = dsMain.ActiveRecord
End If
```



FitToPages Method

-

Decreases the font size of text just enough so that the document will fit on one fewer pages. An error occurs if Word is unable to reduce the page count by one.

expression.**FitToPages**

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

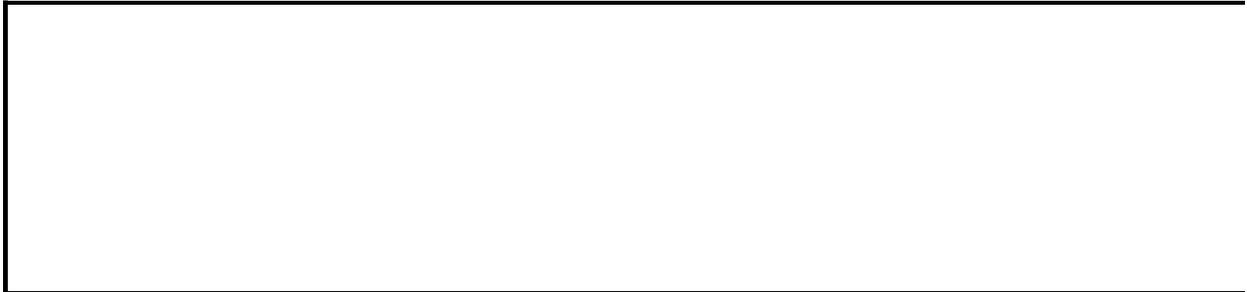
Example

This example attempts to reduce the page count of the active document by one page.

```
On Error GoTo errhandler
ActiveDocument.FitToPages
errhandler:
If Err = 5538 Then MsgBox "Fit to pages failed"
```

This example attempts to reduce the page count of each open document by one page.

```
For Each doc In Documents
    doc.FitToPages
Next doc
```



↳ [Show All](#)

Flip Method

-
Flips a shape horizontally or vertically.

expression.**Flip**(*FlipCmd*)

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

FlipCmd Required [MsoFlipCmd](#). The flip orientation.

MsoFlipCmd can be one of these MsoFlipCmd constants.

msoFlipHorizontal

msoFlipVertical

Example

This example adds a triangle to the active document, duplicates the triangle, and then flips the duplicate triangle vertically and makes it red.

```
Sub FlipShape()  
  With ActiveDocument.Shapes.AddShape( _  
    Type:=msoShapeRightTriangle, Left:=150, _  
    Top:=150, Width:=50, Height:=50).Duplicate  
    .Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)  
    .Flip msoFlipVertical  
  End With  
End Sub
```



Follow Method

Displays a cached document associated with the specified **Hyperlink** object, if it's already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and displays the document in the appropriate application.

Note If the hyperlink uses the file protocol, this method opens the document instead of downloading it.

expression.Follow(NewWindow, AddHistory, ExtraInfo, Method, HeaderInfo)

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

NewWindow Optional **Variant**. **True** to display the target document in a new window. The default value is **False**.

AddHistory Optional **Variant**. This argument is reserved for future use.

ExtraInfo Optional **Variant**. A string or byte array that specifies additional information for HTTP to use to resolve the hyperlink. For example, you can use **ExtraInfo** to specify the coordinates of an image map, the contents of a form, or a FAT file name. The string is either posted or appended, depending on the value of **Method**. Use the [ExtraInfoRequired](#) property to determine whether extra information is required.

Method Optional **Variant**. Specifies the way additional information for HTTP is handled. Can be any [MsoExtraInfoMethod](#) constant.

Enumerated type can be one of these enumerated type constants.

msoMethodGet ExtraInfo is a string that's appended to the address.

msoMethodPost ExtraInfo is posted as a string or a byte array.

HeaderInfo Optional **Variant**. A string that specifies header information for the HTTP request. The default value is an empty string. You can combine several

header lines into a single string by using the following syntax: "*string1*" & **vbCr** & "*string2*". The specified string is automatically converted into ANSI characters. Note that the ***HeaderInfo*** argument may overwrite default HTTP header fields.

Example

This example follows the first hyperlink in Home.doc.

```
Documents("Home.doc").HyperLinks(1).Follow
```

This example inserts a hyperlink to www.msn.com and then follows the hyperlink.

```
Dim hypTemp As Hyperlink
```

```
With Selection
```

```
    .Collapse Direction:=wdCollapseEnd
```

```
    .InsertAfter "MSN "
```

```
    .Previous
```

```
End With
```

```
Set hypTemp = ActiveDocument.Hyperlinks.Add( _
```

```
    Address:="http://www.msn.com", _
```

```
    Anchor:=Selection.Range)
```

```
hypTemp.Follow NewWindow:=False, AddHistory:=True
```



FollowHyperlink Method

Displays a cached document, if it's already been downloaded. Otherwise, this method resolves the hyperlink, downloads the target document, and displays the document in the appropriate application.

Note If the hyperlink uses the file protocol, this method opens the document instead of downloading it.

expression.**FollowHyperlink**(*Address*, *SubAddress*, *NewWindow*, *AddHistory*, *ExtraInfo*, *Method*, *HeaderInfo*)

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

Address Required **String**. The address of the target document.

SubAddress Optional **Variant**. The location within the target document. The default value is an empty string.

NewWindow Optional **Variant**. **True** to display the target location in a new window. The default value is **False**.

AddHistory Optional **Variant**. **True** to add the link to the current day's history folder.

ExtraInfo Optional **Variant**. A string or a byte array that specifies additional information for HTTP to use to resolve the hyperlink. For example, you can use **ExtraInfo** to specify the coordinates of an image map, the contents of a form, or a FAT file name. The string is either posted or appended, depending on the value of **Method**. Use the [ExtraInfoRequired](#) property to determine whether extra information is required.

Method Optional **Variant**. Specifies the way additional information for HTTP is handled. [MsoExtraInfoMethod](#).

Can be one of the following **MsoExtraInfoMethod** constants.

Constant	Description
msoMethodGet	<i>ExtraInfo</i> is a string that's appended to the address.
msoMethodPost	<i>ExtraInfo</i> is posted as a string or a byte array.

HeaderInfo Optional **Variant**. A string that specifies header information for the HTTP request. The default value is an empty string. You can combine several header lines into a single string by using the following syntax: "*string1*" & **vbCr** & "*string2*". The specified string is automatically converted into ANSI characters. Note that the **HeaderInfo** argument may overwrite default HTTP header fields.

Example

This example follows the specified URL address and displays the Microsoft home page in a new window.

```
ActiveDocument.FollowHyperlink _  
    Address:="http://www.Microsoft.com", _  
    NewWindow:=True, AddHistory:=True
```

This example displays the HTML document named "Default.htm."

```
ActiveDocument.FollowHyperlink Address:="file:C:\Pages\Default.htm"
```



Formula Method

-
Inserts an = (Formula) field that contains the specified formula into a table cell.

expression.**Formula**(*Formula*, *NumFormat*)

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

Formula Optional **Variant**. The mathematical formula you want the = (Formula) field to evaluate. Spreadsheet-type references to table cells are valid. For example, "=SUM(A4:C4)" specifies the first three values in the fourth row. For more information about the = (Formula) field, see [Field codes:= \(Formula\) field](#).

NumFormat Optional **Variant**. A format for the result of the = (Formula) field. For information about the types of formats you can apply, see [Numeric Picture \(\#\) field switch](#).

Remarks

Formula is optional as long as there is at least one cell that contains a value above or to the left of the cell that contains the insertion point. If the cells above the insertion point contain values, the inserted field is {=SUM(ABOVE)}; if the cells to the left of the insertion point contain values, the inserted field is {=SUM(LEFT)}. If both the cells above the insertion point and the cells to the left of the insertion point contain values, Microsoft Word uses the following rules to determine which SUM function to insert:

- If the cell immediately above the insertion point contains a value, Word inserts {=SUM(ABOVE)}.
- If the cell immediately above the insertion point doesn't contain a value and the cell immediately to the left of it does, Word inserts {=SUM(LEFT)}.
- If neither adjoining cell contains a value, Word inserts {=SUM(ABOVE)}.
- If you don't specify **Formula** and all the cells above and to the left of the insertion point are empty, the result of the field is an error.

Example

This example creates a 3x3 table at the beginning of the active document and then averages the numbers in the first column.

```
Set myRange = ActiveDocument.Range(0, 0)
Set myTable = ActiveDocument.Tables.Add(myRange, 3, 3)
With myTable
    .Cell(1,1).Range.InsertAfter "100"
    .Cell(2,1).Range.InsertAfter "50"
    .Cell(3,1).Formula Formula:="=Average(Above)"
End With
```

This example inserts a formula at the insertion point that determines the largest number in the cells above the selected cell.

```
Selection.Collapse Direction:=wdCollapseStart
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells(1).Formula Formula:="=Max(Above)"
Else
    MsgBox "The insertion point is not in a table."
End If
```



Forward Method

-

Opens a new e-mail message with an empty **To:** line for forwarding the active message. This method is available only if you are using Word as your e-mail editor.

expression.**Forward**

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

Example

This example opens a new e-mail message for forwarding the active message.

`Application.MailMessage.Forward`



GetAddress Method

Returns an address from the default address book.

expression.**GetAddress**(*Name*, *AddressProperties*, *UseAutoText*, *DisplaySelectDialog*, *SelectDialog*, *CheckNamesDialog*, *RecentAddressesChoice*, *UpdateRecentAddresses*)

expression Required. An expression that returns an **Application** object.

Name Optional **Variant**. The name of the addressee, as specified in the **Search Name** dialog box in the address book.

AddressProperties Optional **Variant**. If *UseAutoText* is **True**, this argument denotes the name of an AutoText entry that defines a sequence of address book properties. If *UseAutoText* is **False** or omitted, this argument defines a custom layout. Valid address book property names or sets of property names are surrounded by angle brackets ("**<**" and "**>**") and separated by a space or a paragraph mark (for example, "<PR_GIVEN_NAME> <PR_SURNAME>" & vbCr & "<PR_OFFICE_TELEPHONE_NUMBER>").

If this argument is omitted, default AutoText entry named "AddressLayout" is used. If "AddressLayout" hasn't been defined, the following address layout definition is used: "<PR_GIVEN_NAME> <PR_SURNAME>" & vbCr & "<PR_STREET_ADDRESS>" & vbCr & "<PR_LOCALITY>" & ", " & "<PR_STATE_OR_PROVINCE>" & " " & "<PR_POSTAL_CODE>" & vbCr & "<PR_COUNTRY>".

For a list of the valid address book property names, see the [AddAddress](#) method.

UseAutoText Optional **Variant**. **True** if *AddressProperties* specifies the name of an AutoText entry that defines a sequence of address book properties; **False** if it specifies a custom layout.

DisplaySelectDialog Optional **Variant**. Specifies whether the **Select Name** dialog box is displayed, as shown in the following table.

Value	Result
0 (zero)	The Select Name dialog box isn't displayed.
1 or omitted	The Select Name dialog box is displayed.
2	The Select Name dialog box isn't displayed, and no search for a specific name is performed. The address returned by this method will be the previously selected address.

SelectDialog Optional **Variant**. Specifies how the **Select Name** dialog box should be displayed (that is, in what mode), as shown in the following table.

Value	Display mode
0 (zero) or omitted	Browse mode
1	Compose mode, with only the To: box
2	Compose mode, with both the To: and CC: boxes

CheckNamesDialog Optional **Variant**. **True** to display the **Check Names** dialog box when the value of the **Name** argument isn't specific enough.

RecentAddressesChoice Optional **Variant**. **True** to use the list of recently used return addresses.

UpdateRecentAddresses Optional **Variant**. **True** to add an address to the list of recently used addresses; **False** to not add the address. If **SelectDialog** is set to 1 or 2, this argument is ignored.

Example

This example sets the variable `strAddress` to John Smith's address, moves the insertion point to the beginning of the document, and inserts the address. The inserted address will include the default address book properties.

```
Dim strAddress

strAddress = Application.GetAddress(Name:="John Smith", _
    CheckNamesDialog:=True)
ActiveDocument.Range(Start:=0, End:=0).InsertAfter strAddress
```

The following example returns John Smith's address, using the "My Address Layout" AutoText entry as the layout definition. "My Address Layout" is defined in the active template and contains a set of address properties assigned to the `text$` variable. The example also adds John Smith's address to the list of recently used addresses.

```
Dim TagIDArray(0 To 3) As String
Dim ValueArray(0 To 3) As String
Dim strAddress As String

TagIDArray(0) = "PR_DISPLAY_NAME"
TagIDArray(1) = "PR_GIVEN_NAME"
TagIDArray(2) = "PR_SURNAME"
TagIDArray(3) = "PR_COMMENT"
ValueArray(0) = "Display_Name"
ValueArray(1) = "John"
ValueArray(2) = "Smith"
ValueArray(3) = "This is a comment"

Application.AddAddress TagID:=TagIDArray(), Value:=ValueArray()
strAddress = Application.GetAddress(Name:="John Smith", _
    UpdateRecentAddresses:=True)
```



GetCrossReferenceItems Method

Returns an array of items that can be cross-referenced based on the specified cross-reference type. The array corresponds to the items listed in the **For which** box in the **Cross-reference** dialog box (**Insert** menu).

Note An item returned by this method can be used as the *ReferenceWhich* argument for the [InsertCrossReference](#) method.

expression.**GetCrossReferenceItems**(*ReferenceType*)

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

ReferenceType Required **Variant**. The type of item you want to insert a cross-reference to. [WdReferenceType](#).

Can be one of the following **WdReferenceType** constants.

wdRefTypeBookmark

wdRefTypeEndnote

wdRefTypeFootnote

wdRefTypeHeading

wdRefTypeNumberedItem.

Example

This example displays the name of the first bookmark in the active document that can be cross-referenced.

```
If ActiveDocument.Bookmarks.Count >= 1 Then
    myBookmarks = ActiveDocument.GetCrossReferenceItems( _
        wdRefTypeBookmark)
```

```
MsgBox myBookmarks(1)
End If
```

This example uses the **GetCrossReferenceItems** method to retrieve a list of headings that can be cross-referenced and then inserts a cross-reference to the page that includes the heading "Introduction."

```
myHeadings = _
    ActiveDocument.GetCrossReferenceItems(wdRefTypeHeading)
For i = 1 To Ubound(myHeadings)
    If Instr(LCase$(myHeadings(i)), "introduction") Then
        Selection.InsertCrossReference _
            ReferenceType:=wdRefTypeHeading, _
            ReferenceKind:=wdPageNumber, ReferenceItem:=i
        Selection.InsertParagraphAfter
    End If
Next i
```



↳ [Show All](#)

GetDefaultTheme Method

Returns a **String** that represents the name of the default [theme](#) plus the theme formatting options Microsoft Word uses for new documents, e-mail messages, or Web pages.

expression.**GetDefaultTheme**(*DocumentType*)

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

DocumentType Required The type of new document for which you want to retrieve the default theme name. [WdDocumentMedium](#).

WdDocumentMedium can be one of these WdDocumentMedium constants.

wdEmailAddress

wdDocument

wdWebPage

Remarks

You can also use the [ThemeName](#) property to return and set the default theme for new e-mail messages.

Example

This example displays the name of the theme Word uses for new Web pages.

```
MsgBox Application.GetDefaultTheme(wdWebPage)
```



GetLetterContent Method

-
Retrieves letter elements from the specified document and returns a [LetterContent](#) object.

expression.**GetLetterContent**

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

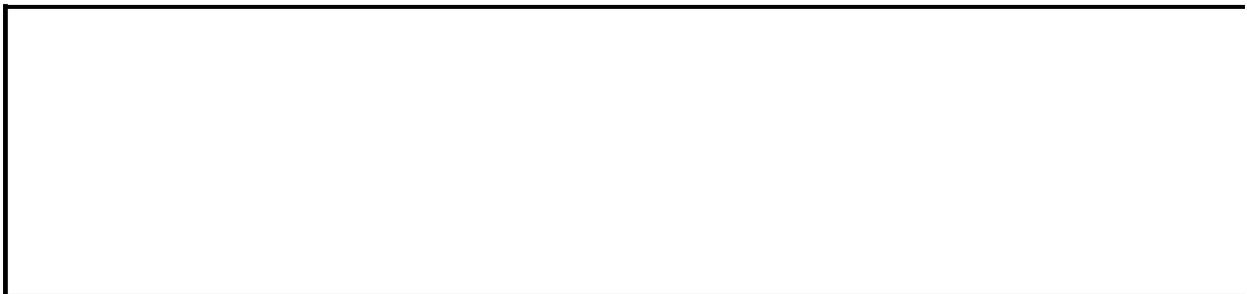
Example

This example displays the salutation and recipient name from the letter in the active document.

```
MsgBox ActiveDocument.GetLetterContent.Salutation _  
    & ActiveDocument.GetLetterContent.RecipientName
```

This example retrieves letter elements from the active document, changes the list of carbon copy (CC) recipients by setting the **CCList** property, and then uses the **SetLetterContent** method to update the active document to reflect the changes.

```
Set myLetterContent = ActiveDocument.GetLetterContent  
With myLetterContent  
    .CCList = "J. Burns, L. Scarpaczyk, K. Wong"  
    .RecipientName = "Amy Anderson"  
    .RecipientAddress = "123 Main" & vbCr & "Bellevue, WA 98004"  
    .LetterStyle = wdFullBlock  
End With  
ActiveDocument.SetLetterContent LetterContent:=myLetterContent
```



GetPoint Method

Returns the screen coordinates of the specified range or shape.

expression.**GetPoint**(**ScreenPixelsLeft**, **ScreenPixelsTop**, **ScreenPixelsWidth**, **ScreenPixelsHeight**, *obj*)

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

ScreenPixelsLeft Required **Long**. The variable name to which you want Microsoft Word to return the value for the left edge of the object.

ScreenPixelsTop Required **Long**. The variable name to which you want Word to return the value for the top edge of the object.

ScreenPixelsWidth Required **Long**. The variable name to which you want Word to return the value for the width of the object.

ScreenPixelsHeight Required **Long**. The variable name to which you want Word to return the value for the height of the object.

obj Required **Object**. A **Range** or **Shape** object.

Remarks

If the entire range or shape isn't visible on the screen, an error occurs.

Example

This example examines the current selection and returns its screen coordinates.

```
Dim pLeft As Long
Dim pTop As Long
Dim pWidth As Long
Dim pHeight As Long

Activewindow.GetPoint pLeft, pTop, pWidth, pHeight, _
    Selection.Range
MsgBox "Left = " & pLeft & vbLf _
    & "Top = " & pTop & vbLf _
    & "Width = " & pWidth & vbLf _
    & "Height = " & pHeight
```



↳ [Show All](#)

GetSpellingSuggestions Method

► [GetSpellingSuggestions method as it applies to the Range object.](#)

Returns a [SpellingSuggestions](#) collection that represents the words suggested as spelling replacements for the first word in the specified range.

expression.**GetSpellingSuggestions**(*CustomDictionary*, *IgnoreUppercase*, *MainDictionary*, *SuggestionMode*, *CustomDictionary2*, *CustomDictionary3*, *CustomDictionary4*, *CustomDictionary5*, *CustomDictionary6*, *CustomDictionary7*, *CustomDictionary8*, *CustomDictionary9*, *CustomDictionary10*)

expression Required. An expression that returns one of the above objects.

CustomDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the custom dictionary.

IgnoreUppercase Optional **Variant**. **True** to ignore words in all uppercase letters. If this argument is omitted, the current value of the **IgnoreUppercase** property is used.

MainDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the main dictionary. If you don't specify a main dictionary, Microsoft Word uses the main dictionary that corresponds to the language formatting of the first word in the range.

SuggestionMode Optional **Variant**. Specifies the way Word makes spelling suggestions. Can be one of the following [WdSpellingWordType](#) constants. The default value is **WdSpellword**.

WdSpellingWordType can be one of these **WdSpellingWordType** constants.

wdAnagram

wdSpellword

wdWildcard

CustomDictionary2 – CustomDictionary10 Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of an additional custom dictionary. You can specify as many as nine additional dictionaries.

► [GetSpellingSuggestions method as it applies to the Application and Global objects.](#)

Returns a [SpellingSuggestions](#) collection that represents the words suggested as spelling replacements for a given word.

expression.**GetSpellingSuggestions**(*Word*, *CustomDictionary*, *IgnoreUppercase*, *MainDictionary*, *SuggestionMode*, *CustomDictionary2*, *CustomDictionary3*, *CustomDictionary4*, *CustomDictionary5*, *CustomDictionary6*, *CustomDictionary7*, *CustomDictionary8*, *CustomDictionary9*, *CustomDictionary10*)

expression Required. An expression that returns one of the above objects.

Word Required **String**. The word whose spelling is to be checked.

CustomDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the custom dictionary.

IgnoreUppercase Optional **Variant**. **True** to ignore words in all uppercase letters. If this argument is omitted, the current value of the **IgnoreUppercase** property is used.

MainDictionary Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of the main dictionary. If you don't specify a main dictionary, Microsoft Word uses the main dictionary that corresponds to the language formatting of **Word** or of the first word in the range.

SuggestionMode Optional **Variant**. Specifies the way Word makes spelling suggestions. Can be one of the following [WdSpellingWordType](#) constants. The default value is **WdSpellword**.

WdSpellingWordType can be one of these **WdSpellingWordType** constants.

wdAnagram

wdSpellword

wdWildcard

CustomDictionary2 – CustomDictionary10 Optional **Variant**. Either an expression that returns a **Dictionary** object or the file name of an additional custom dictionary. You can specify as many as nine additional dictionaries.

Remarks

If the word is spelled correctly, the **Count** property of the **SpellingSuggestions** object returns 0 (zero).

Example

▶ [As it applies to the **Range** object.](#)

This example looks for the alternate spelling suggestions for the first word in the selection. If there are suggestions, the example runs a spelling check on the selection.

```
If Selection.Range.GetSpellingSuggestions.Count = 0 Then
    MsgBox "No suggestions."
Else
    Selection.Range.CheckSpelling
End If
```

▶ [As it applies to the **Global** object.](#)

This example looks for the alternate spelling suggestions for the word "?ook." Suggestions include replacements for the ? wildcard character. Any suggested spellings are displayed in message boxes.

```
Sub DisplaySuggestions()
    Dim sugList As SpellingSuggestions
    Dim sug As SpellingSuggestion
    Dim strSugList As String
    Set sugList = GetSpellingSuggestions(Word:="lrok", _
        SuggestionMode:=wdSpellword)
    If sugList.Count = 0 Then
        MsgBox "No suggestions."
    Else
        For Each sug In sugList
            strSugList = strSugList & vbTab & sug.Name & vbLf
        Next sug
        MsgBox "The suggestions for this word are: " _
            & vbLf & strSugList
    End If
End Sub
```



GoBack Method

-
Moves the insertion point among the last three locations where editing occurred in the active document (the same as pressing SHIFT+F5).

expression.**GoBack**

expression Required. An expression that returns an **Application** object.

Example

This example opens the most recently used file and then moves the insertion point to the location where editing last occurred.

```
RecentFiles(1).Open  
Application.GoBack
```



GoForward Method

-
Moves the insertion point forward among the last three locations where editing occurred in the active document.

expression.**GoForward**

expression Required. An expression that returns an **Application** object.

Example

This example moves the insertion point to the next location where editing occurred.

Application.**GoForward**



↳ [Show All](#)

GoTo Method

Document or **Range** object: Returns a **Range** object that represents the start position of the specified item, such as a page, bookmark, or field.

Selection object: Moves the insertion point to the character position immediately preceding the specified item, and returns a **Range** object (except for the **wdGoToGrammaticalError**, **wdGoToProofreadingError**, or **wdGoToSpellingError** constant).

expression.GoTo(What, Which, Count, Name)

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

What Optional **VARIANT**. The kind of item to which the range or selection is moved. Can be one of the [WdGoToItem](#) constants.

WdGoToItem can be one of these WdGoToItem constants.

wdGoToBookmark

wdGoToComment

wdGoToEndnote

wdGoToEquation

wdGoToField

wdGoToFootnote

wdGoToGrammaticalError

wdGoToGraphic

wdGoToHeading

wdGoToLine

wdGoToObject

wdGoToPage

wdGoToPercent

wdGoToProofreadingError
wdGoToRevision
wdGoToSection
wdGoToSpellingError
wdGoToTable

Which Optional **Variant**. The item to which the range or selection is moved. Can be one of the [WdGoToDirection](#) constants. The following examples are functionally equivalent; they both move the selection to the first heading in the document.

WdGoToDirection can be one of these **WdGoToDirection** constants.

wdGoToAbsolute
wdGoToFirst
wdGoToLast
wdGoToNext
wdGoToPrevious
wdGoToRelative

```
Selection.GoTo What:=wdGoToHeading, Which:=wdGoToFirst  
Selection.GoTo What:=wdGoToHeading, Which:=wdGoToAbsolute, Count:=1
```

Count Optional **Variant**. The number of the item in the document. The default value is 1. The following example moves the selection to the fourth line in the document.

```
Selection.GoTo What:=wdGoToLine, Which:=wdGoToAbsolute, Count:=4
```

Only positive values are valid. To specify an item that precedes the range or selection, use **wdGoToPrevious** as the **Which** argument and specify a **Count** value. The following example moves the selection up two lines.

```
Selection.GoTo What:=wdGoToLine, Which:=wdGoToPrevious, Count:=2
```

Name Optional **Variant**. If the **What** argument is **wdGoToBookmark**, **wdGoToComment**, **wdGoToField**, or **wdGoToObject**, this argument specifies a name. The following example moves to the next DATE field.

```
Selection.GoTo What:=wdGoToField, Name:="Date"
```

Remarks

When you use the **GoTo** method with the **wdGoToGrammaticalError**, **wdGoToProofreadingError**, or **wdGoToSpellingError** constant, the **Range** that's returned includes any grammar error text or spelling error text.

Example

This example moves the selection to the first cell in the next table.

```
Selection.GoTo What:=wdGoToTable, Which:=wdGoToNext
```

This example moves the insertion point just before the fifth endnote reference mark in the active document.

```
If ActiveDocument.Endnotes.Count >= 5 Then
    Selection.GoTo What:=wdGoToEndnote, _
                Which:=wdGoToAbsolute, Count:=5
End If
```

This example sets R1 equal to the first footnote reference mark in the active document.

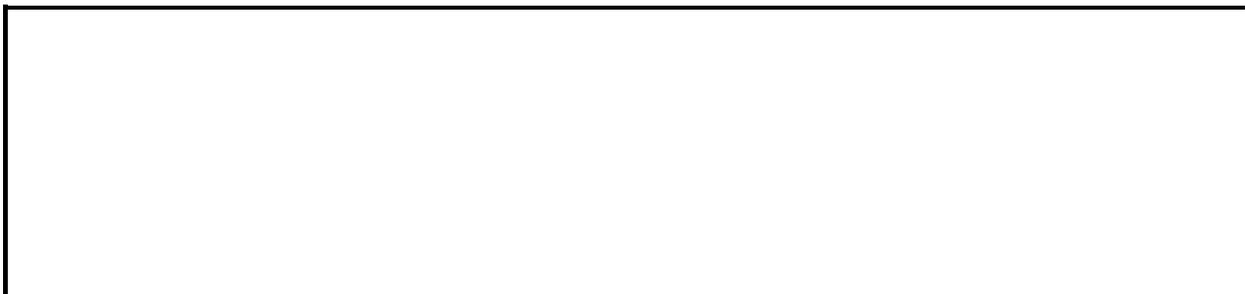
```
If ActiveDocument.Footnotes.Count >= 1 Then
    Set R1 = ActiveDocument.GoTo(What:=wdGoToFootnote, _
                Which:=wdGoToFirst)
    R1.Expand Unit:=wdCharacter
End If
```

This example moves the selection down four lines.

```
Selection.GoTo What:=wdGoToLine, Which:=wdGoToRelative, Count:=4
```

This example moves the selection back two pages.

```
Selection.GoTo What:=wdGoToPage, Which:=wdGoToPrevious, Count:=2
```



↳ [Show All](#)

GoToNext Method

▶ [GoToNext method as it applies to the **Range** and **Selection** objects.](#)

Returns a **Range** object that refers to the start position of the next item or location specified by the *What* argument. If you apply this method to the **Selection** object, the method moves the selection to the specified item (except for the **wdGoToGrammaticalError**, **wdGoToProofreadingError**, and **wdGoToSpellingError** constants). **Range** object.

Note When you use this method with the **wdGoToGrammaticalError**, **wdGoToProofreadingError**, or **wdGoToSpellingError** constant, the **Range** object that's returned includes any grammar error text or spelling error text.

expression.GoToNext(*What*)

expression Required. An expression that returns one of the above objects.

What Required [WdGoToItem](#).

WdGoToItem can be one of these WdGoToItem constants.

wdGoToComment

wdGoToEquation

wdGoToFootnote

wdGoToGraphic

wdGoToLine

wdGoToPage

wdGoToProofreadingError

wdGoToSpellingError

wdGoToBookmark

wdGoToEndnote

wdGoToField

wdGoToGrammaticalError

wdGoToHeading
wdGoToObject
wdGoToPercent
wdGoToSection
wdGoToTable

▶ [GoToNext method as it applies to the MailMessage object.](#)

Displays the next mail message if you are using Word as your e-mail editor.

expression.**GoToNext**

expression Required. An expression that returns one of the above objects.

Example

This example adds a bookmark at the top of page 2 in the active document.

```
Set myRange = ActiveDocument.Words(1).GoToNext(What:=wdGoToPage)
ActiveDocument.Bookmarks.Add Name:="Page2", Range:=myRange
```

This example moves to the next field and selects it.

```
With Selection
    Set myRange = .GoToNext(What:=wdGoToField)
    .MoveRight Unit:=wdWord, Extend:=wdExtend
    .Fields(1).Select
End With
```



↳ [Show All](#)

GoToPrevious Method

▶ [GoToPrevious method as it applies to the **Range** and **Selection** objects.](#)

Returns a **Range** object that refers to the start position of the previous item or location specified by the *What* argument. If applied to a **Selection** object, **GoToPrevious** moves the selection to the specified item. **Range** object.

expression.**GoToPrevious**(*What*)

expression Required. An expression that returns one of the above objects.

What Required The item that the specified range or selection is to be moved to. [WdGoToItem](#).

WdGoToItem can be one of these WdGoToItem constants.

wdGoToComment

wdGoToEquation

wdGoToFootnote

wdGoToGraphic

wdGoToLine

wdGoToPage

wdGoToProofreadingError

wdGoToSpellingError

wdGoToBookmark

wdGoToEndnote

wdGoToField

wdGoToGrammaticalError

wdGoToHeading

wdGoToObject

wdGoToPercent

wdGoToSection

wdGoToTable

- ▶ [GoToPrevious method as it applies to the MailMessage object.](#)

Displays the previous mail message if you are using Word as your e-mail editor.

expression.**GoToPrevious**

expression Required. An expression that returns one of the above objects.

Example

This example moves to the previous field in the active document.

```
Selection.GoToPrevious What:=wdGoToField
```

This example creates a range that references the last footnote reference marker in the active document.

```
Set myRange = ActiveDocument.Words.Last _  
    .GoToPrevious(What:=wdGoToFootnote)  
myRange.Expand Unit:=wdCharacter
```



Group Method

-

Groups the shapes in the specified range. Returns the grouped shapes as a single **Shape** object.

expression.**Group**

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

Remarks

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 myDocument, 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.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    .AddShape(msoShapeCan, 50, 10, 100, 200).Name = "shpOne"
    .AddShape(msoShapeCube, 150, 250, 100, 200).Name = "shpTwo"
    With .Range(Array("shpOne", "shpTwo")).Group
        .Fill.PresetTextured msoTextureBlueTissuePaper
        .Rotation = 45
        .ZOrder msoSendToBack
    End With
End With
```



Grow Method

-

Increases the font size to the next available size. If the selection or range contains more than one font size, each size is increased to the next available setting.

expression.**Grow**

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

Example

This example increases the font size of the fourth word in a new document.

```
Dim rngTemp As Range
```

```
Set rngTemp = Documents.Add.Content
```

```
rngTemp.InsertAfter "This is a test of the Grow method."
```

```
MsgBox "Click OK to increase the font size of the fourth word."
```

```
rngTemp.Words(4).Font.Grow
```

This example increases the font size of the selected text.

```
If Selection.Type = wdSelectionNormal Then
```

```
    Selection.Font.Grow
```

```
Else
```

```
    MsgBox "You need to select some text."
```

```
End If
```



Help Method

-
Displays on-line Help information.

expression.**Help**(*HelpType*)

expression An expression that returns a **Application** object.

HelpType Required **Variant**. The on-line Help topic or window. Can be any of these [WdHelpType](#) constants.

Enumerated type can be one of these enumerated type constants.

wdHelp Displays the **Help Topics** dialog box.

wdHelpAbout Displays the **About Microsoft Word** dialog box (**Help** menu).

wdHelpActiveWindow Displays Help describing the command associated with the active view or pane.

wdHelpContents Displays the **Help Topics** dialog box.

wdHelpHWP Displays Help topics for AreA Hangul users.

wdHelpIchitaro Displays Help topics for Ichitaro users.

wdHelpIndex Displays the Help Topics dialog box.

wdHelpPE2 Displays Help topics for IBM Personal Editor 2 users.

wdHelpPSSHelp Displays product support information.

wdHelpSearch Displays the **Help Topics** dialog box.

wdHelpUsingHelp Displays a list of Help topics that describe how to use Help.

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.

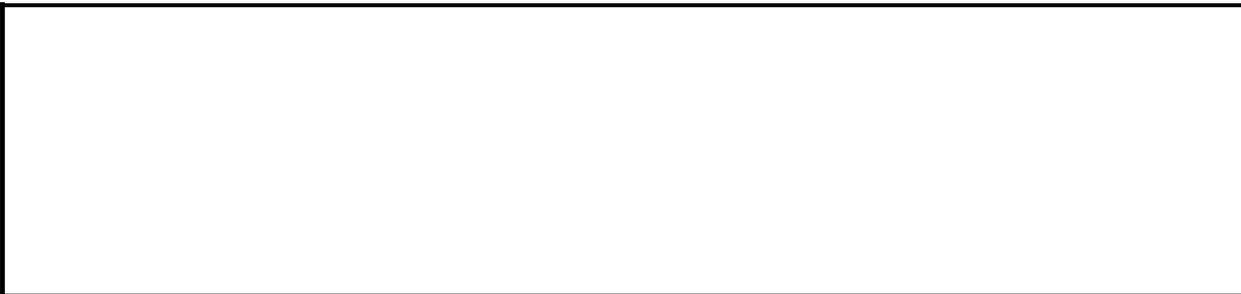
Example

This example displays the Help Topics dialog box.

```
Help HelpType:=wdHelp
```

This example displays a list of Help topics that describe how to use Help.

```
Help HelpType:=wdHelpUsingHelp
```



HelpTool Method

-

Changes the pointer from an arrow to a question mark, indicating that you'll get context-sensitive Help information about the next command or screen element you click. If you click text, Word displays a box describing current paragraph and character formats. Pressing ESC turns the pointer back to an arrow.

expression.**HelpTool()**

expression Required. An expression that returns an **Application** object.

Example

This example changes the mouse pointer from an arrow to a question mark.

`Application.HelpTool`



HomeKey Method

Moves or extends the selection to the beginning of the specified unit. This method returns an integer that indicates the number of characters the selection was actually moved, or it returns 0 (zero) if the move was unsuccessful.

Note This method corresponds to functionality of the HOME key.

expression.**HomeKey**(*Unit*, *Extend*)

expression An expression that returns a **Selection** object.

Unit Optional **Variant**. The unit by which the selection is to be moved or extended. [WdUnits](#).

Can be one of the following **WdUnits** constants.

wdStory

wdColumn

wdLine

wdRow. The default value is **wdLine**.

Extend Optional **Variant**. Specifies the way the selection is moved. [WdMovementType](#).

Can be one of the following **WdMovementType** constants.

wdMove

wdExtend

If the value of this argument is **wdMove**, the selection is collapsed to an insertion point and moved to the beginning of the specified unit. If it's

wdExtend, the beginning of the selection is extended to the beginning of the specified unit. The default value is **wdMove**.

Example

This example moves the selection to the beginning of the current story. If the selection is in the main text story, the selection is moved to the beginning of the document.

```
Selection.HomeKey Unit:=wdStory, Extend:=wdMove
```

This example moves the selection to the beginning of the current line and assigns the number of characters moved to the pos variable.

```
pos = Selection.HomeKey(Unit:=wdLine, Extend:=wdMove)  
If pos = 0 Then StatusBar = "Selection was not moved"
```



HTMLDivisionParent Method

Returns an [HTMLDivision](#) object that represents a parent division of the current HTML division.

expression.HTMLDivisionParent(*LevelsUp*)

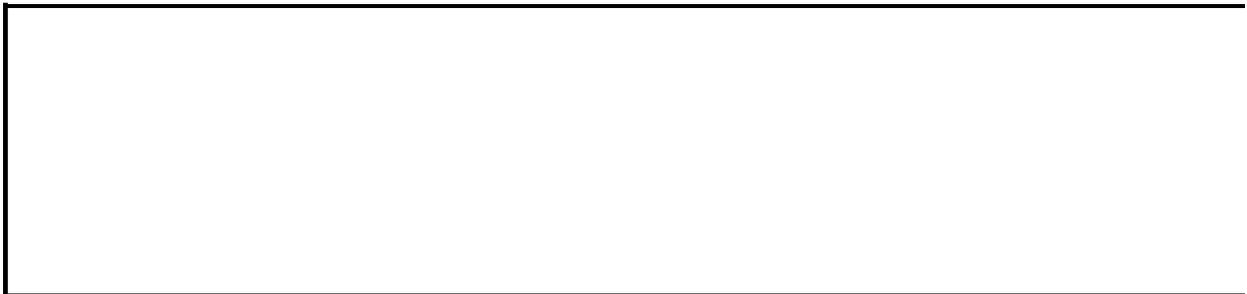
expression Required. An expression that returns one of the objects in the Applies To list.

LevelsUp Optional **Long**. The number of parent divisions to count back to return the desired division. If the **LevelsUp** argument is omitted, the HTML division returned is one level up from the current HTML division.

Example

This example formats the borders for two HTML divisions in the active document. This example assumes that the active document is an HTML document with at least two divisions.

```
Sub FormatHTMLDivisions()  
  With ActiveDocument.HTMLDivisions(1)  
    With .HTMLDivisions(1)  
      .LeftIndent = InchesToPoints(1)  
      .RightIndent = InchesToPoints(1)  
      With .Borders(wdBorderLeft)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .Borders(wdBorderRight)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .HTMLDivisionParent  
        .LeftIndent = InchesToPoints(1)  
        .RightIndent = InchesToPoints(1)  
        With .Borders(wdBorderTop)  
          .Color = wdColorBlack  
          .LineStyle = wdLineStyleDot  
        End With  
        With .Borders(wdBorderBottom)  
          .Color = wdColorBlack  
          .LineStyle = wdLineStyleDot  
        End With  
      End With  
    End With  
  End With  
End Sub
```



InchesToPoints Method

-

Converts a measurement from inches to points (1 inch = 72 points). Returns the converted measurement as a **Single**.

expression.**InchesToPoints**(*Inches*)

expression Optional. An expression that returns an **Application** object.

Inches Required **Single**. The inch value to be converted to points.

Example

This example sets the space before for the selected paragraphs to 0.25 inch.

```
Selection.ParagraphFormat.SpaceBefore = InchesToPoints(0.25)
```

This example prints each open document after setting the left and right margins to 0.65 inch.

```
Dim docLoop As Document
```

```
For Each docLoop in Documents
```

```
    With docLoop
```

```
        .PageSetup.LeftMargin = InchesToPoints(0.65)
```

```
        .PageSetup.RightMargin = InchesToPoints(0.65)
```

```
        .PrintOut
```

```
    End With
```

```
Next docLoop
```



IncreaseSpacing Method

-
Increases the spacing before and after paragraphs in six-point increments.

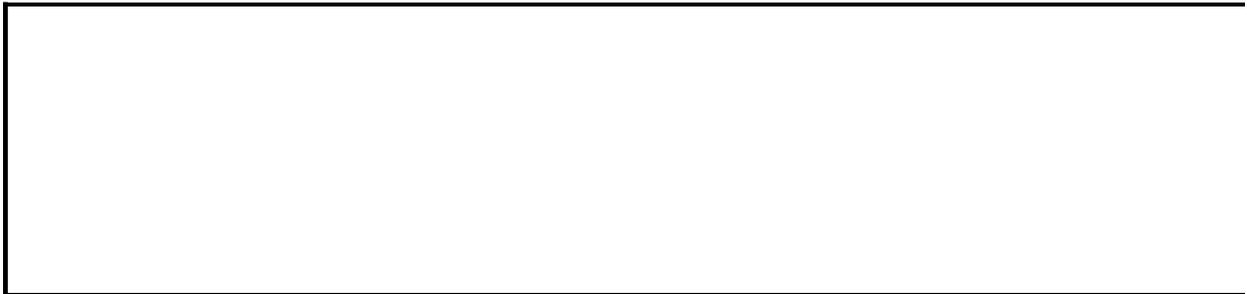
expression.**IncreaseSpacing**

expression Required. An expression that returns a [Paragraphs](#) object.

Example

This example increases the before and after spacing of a paragraph or selection of paragraphs by six points each time the procedure is run.

```
Sub IncreaseParaSpacing()  
    Selection.Paragraphs.IncreaseSpacing  
End Sub
```



IncrementBrightness Method

-
Changes the brightness of the picture by the specified amount. Use the [Brightness](#) property to set the absolute brightness of the picture.

expression.IncrementBrightness(*Increment*)

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

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.

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.

Example

This example creates a duplicate of the first shape on the active document and then moves and darkens the duplicate. For the example to work, the first shape must be either a picture or an OLE object.

```
Dim docActive As Document
Set docActive = ActiveDocument
With docActive.Shapes(1).Duplicate
    .PictureFormat.IncrementBrightness -0.2
    .IncrementLeft 50
    .IncrementTop 50
End With
```



IncrementContrast Method

-
Changes the contrast of the picture by the specified amount. Use the [Contrast](#) property to set the absolute contrast for the picture.

expression.**IncrementContrast**(*Increment*)

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

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.

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.

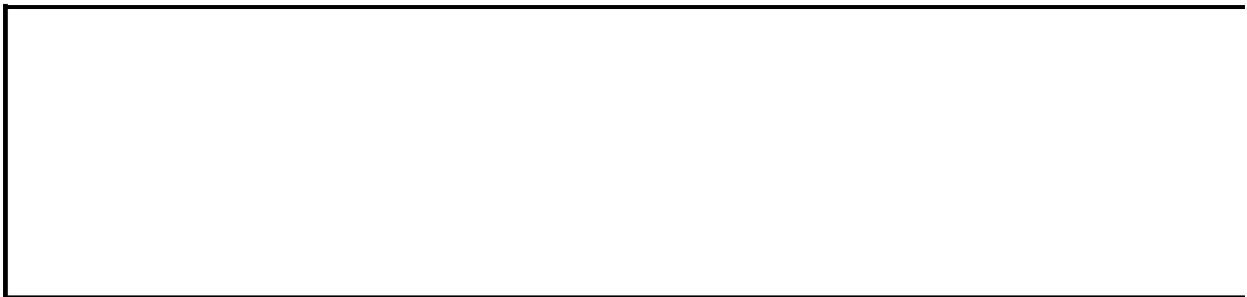
Example

This example increases the contrast for all embedded OLE objects on the active document that aren't already set to maximum contrast.

```
Dim docActive As Document
Dim shapeLoop As Shape

Set docActive = ActiveDocument

For Each shapeLoop In docActive.Shapes
    If shapeLoop.Type = msoEmbeddedOLEObject Then
        shapeLoop.PictureFormat.IncrementContrast 0.1
    End If
Next shapeLoop
```



IncrementLeft Method

Moves the specified shape horizontally by the specified number of points.

expression.**IncrementLeft**(*Increment*)

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

Increment Required **Single**. Specifies how far the shape is to be moved horizontally, in points. A positive value moves the shape to the right; a negative value moves it to the left.

Example

This example duplicates shape one on myDocument, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```



IncrementOffsetX Method

-

Changes the horizontal offset of the shadow by the specified number of points. Use the [OffsetX](#) property to set the absolute horizontal shadow offset.

expression.**IncrementOffsetX**(*Increment*)

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

Increment Required **Single**. Specifies how far the shadow offset is to be moved horizontally, in points. A positive value moves the shadow to the right; a negative value moves it to the left.

Example

This example moves the shadow on the third shape in the active document to the left by 3 points.

```
ActiveDocument.Shapes(3).Shadow.IncrementOffsetX -3
```



IncrementOffsetY Method

-

Changes the vertical offset of the shadow by the specified number of points. Use the [OffsetY](#) property to set the absolute vertical shadow offset.

expression.**IncrementOffsetY**(*Increment*)

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

Increment Required **Single**. Specifies how far the shadow offset is to be moved vertically, in points. A positive value moves the shadow down; a negative value moves it up.

Example

This example moves the shadow on the third shape in the active document up by 3 points.

```
ActiveDocument.Shapes(3).Shadow.IncrementOffsetY -3
```



IncrementRotation Method

-

Changes the rotation of the specified shape around the z-axis by the specified number of degrees. Use the [Rotation](#) property to set the absolute rotation of the shape.

expression.IncrementRotation(*Increment*)

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

Increment Required **Single**. Specifies how far the shape is to be rotated horizontally, in degrees. A positive value rotates the shape clockwise; a negative value rotates it counterclockwise.

Remarks

To rotate a three-dimensional shape around the x-axis or the y-axis, use the [IncrementRotationX](#) method or the [IncrementRotationY](#) method.

Example

This example duplicates shape one on myDocument, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```



IncrementRotationX Method

-

Changes the rotation of the specified shape around the x-axis by the specified number of degrees. Use the [RotationX](#) property to set the absolute rotation of the shape around the x-axis.

expression.**IncrementRotationX**(*Increment*)

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

Increment Required **Single**. Specifies how much (in degrees) the rotation of the shape around the x-axis is to be changed. Can be a value from -90 through 90. A positive value tilts the shape up; a negative value tilts it down.

Remarks

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, use the [IncrementRotationY](#) method. To change the rotation around the z-axis, use the [IncrementRotation](#) method.

Example

This example tilts the first shape on the active document up 10 degrees. The first shape must be an extruded shape for you to see the effect of this code.

```
ActiveDocument.Shapes(1).ThreeD.IncrementRotationX 10
```



IncrementRotationY Method

-

Changes the rotation of the specified shape around the y-axis by the specified number of degrees. Use the [RotationY](#) property to set the absolute rotation of the shape around the y-axis.

expression.**IncrementRotationY**(*Increment*)

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

Increment Required **Single**. Specifies how much (in degrees) the rotation of the shape around the y-axis is to be changed. 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

To change the rotation of a shape around the x-axis, use the [IncrementRotationX](#) method. To change the rotation around the z-axis, use the [IncrementRotation](#) method.

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.

Example

This example tilts the first shape on the active document 10 degrees to the right. The first shape must be an extruded shape for you to see the effect of this code.

```
ActiveDocument.Shapes(1).ThreeD.IncrementRotationY -10
```



IncrementTop Method

Moves the specified shape vertically by the specified number of points.

expression.**IncrementTop**(*Increment*)

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

Increment Required **Single**. Specifies how far the shape object is to be moved vertically, in points. A positive value moves the shape down; a negative value moves it up.

Example

This example duplicates shape one on myDocument, sets the fill for the duplicate, moves it 70 points to the right and 50 points up, and rotates it 30 degrees clockwise.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```



Indent Method

-

Indents one or more paragraphs by one level.

Note Using this method is equivalent to clicking the **Increase Indent** button on the **Formatting** toolbar.

expression.**Indent**

expression Required. An expression that returns a **Paragraph** or **Paragraphs** object.

Example

This example indents all the paragraphs in the active document twice, and then it removes one level of the indent for the first paragraph.

```
With ActiveDocument.Paragraphs
```

```
    .Indent
```

```
    .Indent
```

```
End With
```

```
ActiveDocument.Paragraphs(1).Outdent
```



IndentCharWidth Method

-
Indents one or more paragraphs by a specified number of characters.

expression.**IndentCharWidth**(*Count*)

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Count Required **Integer**. The number of characters by which the specified paragraphs are to be indented.

Remarks

Using this method is equivalent to clicking the **Increase Indent** button on the **Formatting** toolbar.

Example

This example indents the first paragraph of the active document by 10 characters.

```
With ActiveDocument.Paragraphs(1)  
    .IndentCharWidth 10  
End With
```



IndentFirstLineCharWidth Method

-

Indents the first line of one or more paragraphs by a specified number of characters.

expression.**IndentFirstLineCharWidth**(*Count*)

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Count Required **Integer**. The number of characters by which the first line of each specified paragraph is to be indented.

Example

This example indents the first line of the first paragraph in the active document by 10 characters.

```
With ActiveDocument.Paragraphs(1)  
    .IndentFirstLineCharWidth 10  
End With
```



↳ [Show All](#)

InRange Method

Returns **True** if the range or selection to which the method is applied is contained in the range specified by the **Range** argument.

expression.**InRange**(**Range**)

expression Required. An expression that returns a **Range** or **Selection** object.

Range Required **Range** object. The range to which you want to compare *expression*.

Remarks

This method determines whether the range or selection returned by *expression* is contained in the specified **Range** by comparing the starting and ending character positions, as well as the [story](#) type.

Example

This example determines whether the selection is contained in the first paragraph in the active document.

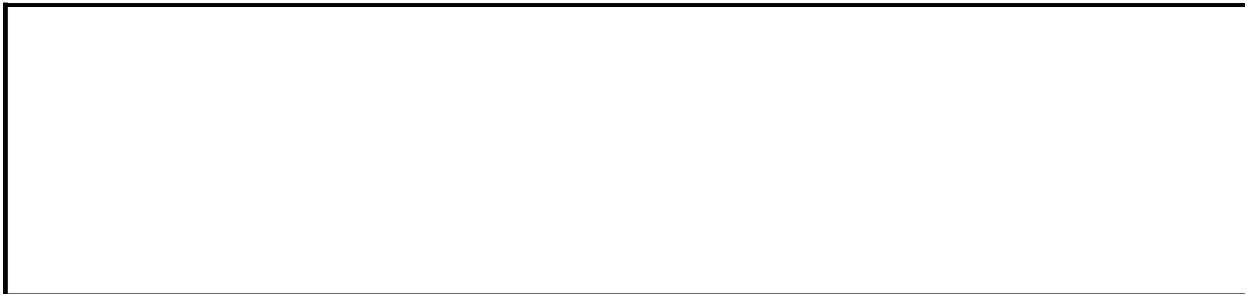
```
status = Selection.InRange(ActiveDocument.Paragraphs(1).Range)
```

This example sets myRange equal to the first word in the active document. If myRange isn't contained in the selection, myRange is selected.

```
Set myRange = ActiveDocument.Words(1)  
If myRange.InRange(Selection.Range) = False Then myRange.Select
```

This example displays a message if the selection is in the footnote story.

```
If Selection.InRange(ActiveDocument _  
    .StoryRanges(wdFootnotesStory)) Then  
    MsgBox "Selection in footnotes"  
End If
```



↳ [Show All](#)

Insert Method

▶ [Insert method as it applies to the **AutoTextEntry** object.](#)

Inserts the AutoText entry in place of the specified range. Returns a **Range** object that represents the AutoText entry.

expression.**Insert**(*Where*, *RichText*)

expression Required. An expression that returns an **AutoTextEntry** object.

Where Required **Range** object. The location for the AutoText entry.

RichText Optional **Variant**. **True** to insert the AutoText entry with its original formatting.

Remarks

If you don't want to replace the range, use the [Collapse](#) method before using this method.

► [Insert method as it applies to the Envelope object.](#)

Inserts an envelope as a separate section at the beginning of the specified document.

expression.Insert(ExtractAddress, Address, AutoText, OmitReturnAddress, ReturnAddress, ReturnAutoText, PrintBarCode, PrintFIMA, Size, Height, Width, FeedSource, AddressFromLeft, AddressFromTop, ReturnAddressFromLeft, ReturnAddressFromTop, DefaultFaceUp, DefaultOrientation, PrintEPostage, Vertical, RecipientNamefromLeft, RecipientNamefromTop, RecipientPostalfromLeft, RecipientPostalfromTop, SenderNamefromLeft, SenderNamefromTop, SenderPostalfromLeft, SenderPostalfromTop)

expression Required. An expression that returns an **Envelope** object.

ExtractAddress Optional **Variant**. **True** to use the text marked by the EnvelopeAddress bookmark (a user-defined bookmark) as the recipient's address.

Address Optional **Variant**. A string that specifies the recipient's address (ignored if **ExtractAddress** is **True**).

AutoText Optional **Variant**. A string that specifies an AutoText entry to use for the address. If specified, **Address** is ignored.

OmitReturnAddress Optional **Variant**. **True** to not insert a return address.

ReturnAddress Optional **Variant**. A string that specifies the return address.

ReturnAutoText Optional **Variant**. A string that specifies an AutoText entry to use for the return address. If specified, **ReturnAddress** is ignored.

PrintBarCode Optional **Variant**. **True** to add a POSTNET bar code. For U.S.

mail only.

PrintFIMA Optional **VARIANT**. **True** to add a Facing Identification Mark (FIMA) for use in presorting courtesy reply mail. For U.S. mail only.

Size Optional **VARIANT**. A string that specifies the envelope size. The string must match one of the sizes listed in the **Envelope size** box in the **Envelope Options** dialog box (for example, "Size 10" or "C4").

Height Optional **VARIANT**. The height of the envelope, measured in points, when the **Size** argument is set to "Custom size."

Width Optional **VARIANT**. The width of the envelope, measured in points, when the **Size** argument is set to "Custom size."

FeedSource Optional **VARIANT**. **True** to use the **FeedSource** property of the **Envelope** object to specify which paper tray to use when printing the envelope.

AddressFromLeft Optional **VARIANT**. The distance, measured in points, between the left edge of the envelope and the recipient's address.

AddressFromTop Optional **VARIANT**. The distance, measured in points, between the top edge of the envelope and the recipient's address.

ReturnAddressFromLeft Optional **VARIANT**. The distance, measured in points, between the left edge of the envelope and the return address.

ReturnAddressFromTop Optional **VARIANT**. The distance, measured in points, between the top edge of the envelope and the return address.

DefaultFaceUp Optional **VARIANT**. **True** to print the envelope face up, **False** to print it face down.

DefaultOrientation Optional **VARIANT**. The orientation for the envelope. Can be any [WdEnvelopeOrientation](#) constant.

wdLeftPortrait

wdCenterPortrait

wdRightPortrait

wdLeftLandscape

wdCenterLandscape

wdRightLandscape

wdLeftClockwise

wdCenterClockwise

wdRightClockwise

PrintEPostage Optional **Variant**. **True** to insert postage from an Internet postage vendor.

Vertical Optional **Variant**. **True** to print vertical text on the envelope. Used for Asian envelopes. Default is **False**.

RecipientNamefromLeft Optional **Variant**. Position of the recipient's name, measured in points from the left edge of the envelope. Used for Asian envelopes.

RecipientNamefromTop Optional **Variant**. Position of the recipient's name, measured in points from the top edge of the envelope. Used for Asian envelopes.

RecipientPostalfromLeft Optional **Variant**. Position of the recipient's postal code, measured in points from the left edge of the envelope. Used for Asian envelopes.

RecipientPostalfromTop Optional **Variant**. Position of the recipient's postal code, measured in points from the top edge of the envelope. Used for Asian envelopes.

SenderNamefromLeft Optional **Variant**. Position of the sender's name, measured in points from the left edge of the envelope. Used for Asian envelopes.

SenderNamefromTop Optional **Variant**. Position of the sender's name, measured in points from the top edge of the envelope. Used for Asian envelopes.

SenderPostalfromLeft Optional **Variant**. Position of the sender's postal code, measured in points from the left edge of the envelope. Used for Asian envelopes.

SenderPostalfromTop Optional **Variant**. Position of the sender's postal code, measured in points from the top edge of the envelope. Used for Asian envelopes.

► [Insert method as it applies to the **ShapeNodes** object.](#)

Inserts a node into a freeform shape.

expression.Insert(Index, SegmentType, EditingType, X1, Y1, X2, Y2, X3, Y3)

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

Index Required **Long**. The number of the shape node after which to insert a new node.

SegmentType Required [MsoSegmentType](#). The type of line that connects the inserted node to the neighboring nodes.

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentLine

msoSegmentCurve

EditingType Required [MsoEditingType](#). The editing property of the inserted node.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto

msoEditingCorner

msoEditingSmooth

msoEditingSymmetric

X1 Required **Single**. If the **EditingType** of the new segment is **msoEditingAuto**, this argument specifies the horizontal distance, measured in points, from the upper-left corner of the document to the end point of the new segment. If the **EditingType** of the new node is **msoEditingCorner**, this argument specifies the horizontal distance, measured in points, from the upper-left corner of the document to the first control point for the new segment.

Y1 Required **Single**. If the **EditingType** of the new segment is **msoEditingAuto**, this argument specifies the vertical distance, measured in points, from the upper-left corner of the document to the end point of the new segment. If the **EditingType** of the new node is **msoEditingCorner**, this argument specifies the vertical distance, measured in points, from the upper-left corner of the document to the first control point for the new segment.

X2 Optional **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance, measured in points, from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the vertical distance, measured in points, from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the horizontal distance, measured in points, from the upper-left corner of the document 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 **Single**. If the *EditingType* of the new segment is **msoEditingCorner**, this argument specifies the vertical distance, measured in points, from the upper-left corner of the document 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.

Example

▶ [As it applies to the **AutoTextEntry** object.](#)

This example inserts the formatted AutoText entry named "one" after the selection.

```
Sub InsertAutoTextEntry()  
    ActiveDocument.Content.Select  
    Selection.Collapse Direction:=wdCollapseEnd  
    ActiveDocument.AttachedTemplate.AutoTextEntries("one").Insert _  
        Where:=Selection.Range, RichText:=True  
End Sub
```

▶ [As it applies to the **Envelope** object.](#)

This example adds a Size 10 envelope to the active document by using the addresses stored in the strAddr and strReturnAddr variables.

```
Sub InsertEnvelope()  
    Dim strAddr As String  
    Dim strReturnAddr As String  
    strAddr = "Max Benson" & vbCr & "123 Skye St." _  
        & vbCr & "OurTown, WA 98107"  
    strReturnAddr = "Paul Borm" & vbCr & "456 Erde Lane" _  
        & vbCr & "OurTown, WA 98107"  
    ActiveDocument.Envelope.Insert Address:=strAddr, _  
        ReturnAddress:=strReturnAddr, Size:="Size 10"  
End Sub
```

▶ [As it applies to the **ShapeNodes** object.](#)

This example selects the third shape in the active document, checks whether the shape is a **Freeform** object, and if it is, inserts a node.

```
Sub InsertShapeNode()  
    ActiveDocument.Shapes(3).Select  
    With Selection.ShapeRange  
        If .Type = msoFreeform Then  
            .Nodes.Insert _  
                Index:=3, SegmentType:=msoSegmentCurve, _  
                EditingType:=msoEditingSymmetric, x1:=35, y1:=100  
        End If  
    End With  
End Sub
```

```
        .Fill.ForeColor.RGB = RGB(0, 0, 200)
        .Fill.Visible = msoTrue
    Else
        MsgBox "This shape is not a Freeform object."
    End If
End With
End Sub
```



InsertAfter Method

-

Inserts the specified text at the end of a range or selection. After this method is applied, the range or selection expands to include the new text.

expression.**InsertAfter**(*Text*)

expression Required. An expression that returns a **Selection** or **Range** object.

Text Required **String**. The text to be inserted.

Remarks

You can insert characters such as quotation marks, tab characters, and nonbreaking hyphens by using the Visual Basic **Chr** function with the **InsertAfter** method. You can also use the following Visual Basic constants: **vbCr**, **vbLf**, **vbCrLf** and **vbTab**.

If you use this method with a range or selection that refers to an entire paragraph, the text is inserted after the ending paragraph mark (the text will appear at the beginning of the next paragraph). To insert text at the end of a paragraph, determine the ending point and subtract 1 from this location (the paragraph mark is one character), as shown in the following example.

```
Set doc = ActiveDocument
Set rngRange = _
    doc.Range(doc.Paragraphs(1).Start, _
    doc.Paragraphs(1).End - 1)
rngRange.InsertAfter _
    " This is now the last sentence in paragraph one."
```

However, if the range or selection ends with a paragraph mark that also happens to be the end of the document, Microsoft Word inserts the text before the final paragraph mark rather than creating a new paragraph at the end of the document.

Also, if the range or selection is a bookmark, Word inserts the specified text but does not extend the range or selection or the bookmark to include the new text.

Example

This example inserts text at the end of the active document. The **Content** property returns a **Range** object.

```
ActiveDocument.Content.InsertAfter "end of document"
```

This example inserts text at the end of the selection and then collapses the selection to an insertion point.

```
With Selection
    .InsertAfter "appended text"
    .Collapse Direction:=wdCollapseEnd
End With
```

This example inserts text from an input box as the second paragraph in the active document.

```
response = InputBox("Type some text")
With ActiveDocument.Paragraphs(1).Range
    .InsertAfter "1." & Chr(9) & response
    .InsertParagraphAfter
End With
```

InsertAutoText Method

-

Attempts to match the text in the specified range or the text surrounding the range with an existing AutoText entry name. If any such match is found, **InsertAutoText** inserts the AutoText entry to replace that text. If a match cannot be found, an error occurs.

Note You can use the [Insert](#) method with an **AutoTextEntry** object to insert a specific AutoText entry.

expression.**InsertAutoText**

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

Example

This example inserts an AutoText entry that matches the text around a selection.

```
Documents.Add  
Selection.TypeText "Best w"  
Selection.Range.InsertAutoText
```

This example inserts an AutoText entry with a name that matches the first word in the active document.

```
Documents.Add  
Selection.TypeText "In "  
Set myRange = ActiveDocument.Words(1)  
myRange.InsertAutoText
```



InsertBefore Method

-

Inserts the specified text before the specified selection or range. After the text is inserted, the selection or range is expanded to include the new text. If the selection or range is a bookmark, the bookmark is also expanded to include the next text.

expression.**InsertBefore**(*Text*)

expression Required. An expression that returns a **Range** or **Selection** object.

Text Required **String**. The text to be inserted.

Remarks

You can insert characters such as quotation marks, tab characters, and nonbreaking hyphens by using the Visual Basic **Chr** function with the **InsertBefore** method. You can also use the following Visual Basic constants: **vbCr**, **vbLf**, **vbCrLf** and **vbTab**.

Example

This example inserts the text "Hamlet" (enclosed in quotation marks) before the selection and then collapses the selection.

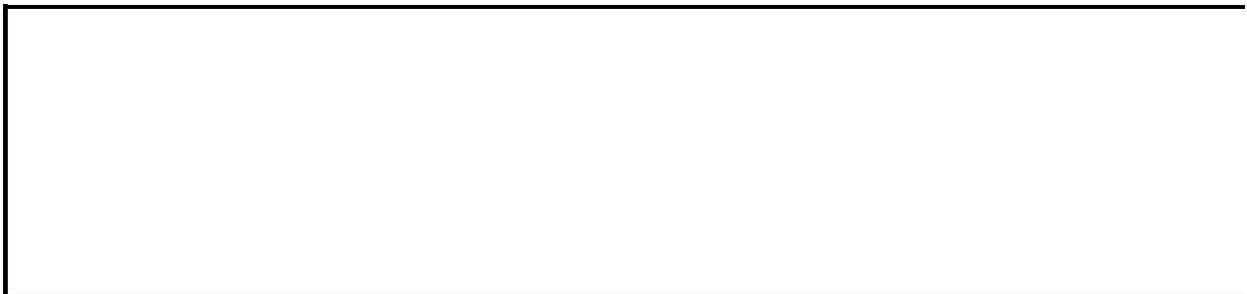
```
With Selection
    .InsertBefore Chr(34) & "Hamlet" & Chr(34) & Chr(32)
    .Collapse Direction:=wdCollapseEnd
End With
```

This example inserts the text "Introduction" as a separate paragraph at the beginning of the active document.

```
With ActiveDocument.Content
    .InsertParagraphBefore
    .InsertBefore "Introduction"
End With
```

This example inserts all the font names in the **FontNames** collection into a new document.

```
Documents.Add
For Each aFont In FontNames
    With Selection
        .InsertBefore aFont
        .Collapse Direction:=wdCollapseEnd
        .TypeParagraph
    End With
Next aFont
```



InsertBreak Method

-
Inserts a page, column, or section break.

expression.**InsertBreak**(*Type*)

expression Required. An expression that returns a **Range** or **Selection** object.

Type Optional **Variant**. The type of break to be inserted. [WdBreakType](#).

Can be one of the following **WdBreakType** constants.

wdPageBreak

wdColumnBreak

wdSectionBreakNextPage

wdSectionBreakContinuous

wdSectionBreakEvenPage

wdSectionBreakOddPage

wdLineBreak

wdLineBreakClearLeft

wdLineBreakClearRight

wdTextWrappingBreak

The default value is **wdPageBreak**.

Remarks

When you insert a page or column break, the range or selection is replaced by the break. If you don't want to replace the range or selection, use the **Collapse** method before using the **InsertBreak** method. When you insert a section break, the break is inserted immediately preceding the **Range** or **Selection** object.

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 inserts a continuous section break immediately preceding the selection.

```
Selection.InsertBreak Type:=wdSectionBreakContinuous
```

This example inserts a page break immediately following the second paragraph in the active document.

```
Set myRange = ActiveDocument.Paragraphs(2).Range  
With myRange  
    .Collapse Direction:=wdCollapseEnd  
    .InsertBreak Type:=wdPageBreak  
End With
```



InsertCaption Method

Inserts a caption immediately preceding or following the specified range or selection.

expression.**InsertCaption**(*Label*, *Title*, *TitleAutoText*, *Position*)

expression Required. An expression that returns a **Range** or **Selection** object.

Label Required **Variant**. The caption label to be inserted. [WdCaptionLabelID](#)

Can be a string or one of the following **WdCaptionLabelID** constants.

wdCaptionEquation

wdCaptionFigure

wdCaptionTable

If the label hasn't yet been defined, an error occurs. Use the **Add** method with the **CaptionLabels** object to define new caption labels.

Title Optional **Variant**. The string to be inserted immediately following the label in the caption (ignored if *TitleAutoText* is specified).

TitleAutoText Optional **Variant**. The AutoText entry whose contents you want to insert immediately following the label in the caption (overrides any text specified by *Title*).

Position Optional **Variant**. Specifies whether the caption will be inserted above or below the **Selection** or **Range** object. [WdCaptionPosition](#)

Can be either of the following **WdCaptionPosition** constants.

wdCaptionPositionAbove

wdCaptionPositionBelow.

Example

This example inserts a caption below the first table in the active document.

```
ActiveDocument.Tables(1).Range.InsertCaption _  
    Label:=wdCaptionTable, _  
    Position:=wdCaptionPositionBelow
```

This example inserts a Figure caption at the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart  
Selection.InsertCaption Label:"Figure", _  
    Title:": Sales Results", Position:=wdCaptionPositionBelow
```



InsertCells Method

-

Adds cells to an existing table. The number of cells inserted is equal to the number of cells in the selection.

Note You can also insert cells by using the [Add](#) method of the **Cells** object.

expression.InsertCells(ShiftCells)

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

ShiftCells Optional [WdInsertCells](#).

Can be one of the following **WdInsertCells** constants.

Constant	Description
wdInsertCellsEntireColumn	Inserts an entire column to the left of the column that contains the selection.
wdInsertCellsEntireRow	Inserts an entire row above the row that contains the selection.
wdInsertCellsShiftDown	Inserts new cells above the selected cells.
wdInsertCellsShiftRight	Insert new cells to the left of the selected cells.

Example

This example inserts new cells to the left of the selected cells, and then it surrounds the selected cells with a red, single-line border.

```
If Selection.Cells.Count >= 1 Then
    Selection.InsertCells ShiftCells:=wdInsertCellsShiftRight
    For Each aBorder In Selection.Borders
        aBorder.LineStyle = wdLineStyleSingle
        aBorder.ColorIndex = wdRed
    Next aBorder
End If
```



InsertColumns Method

-

Inserts columns to the left of the column that contains the selection. If the selection isn't in a table, an error occurs.

Note The number of columns inserted is equal to the number of columns selected. You can also insert columns by using the [Add](#) method of the **Columns** object.

expression.**InsertColumns**

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

Example

This example inserts new columns to the left of the column that contains the selection. The number of columns inserted is equal to the number of columns selected.

```
If Selection.Information(wdWithInTable) = True Then
    With Selection
        .InsertColumns
        .Shading.Texture = wdTexture10Percent
    End With
End If
```



InsertColumnsRight Method

-
Inserts columns to the right of the current selection.

expression.**InsertColumnsRight**

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

Remarks

Microsoft Word inserts as many columns as there are in the current selection.

In order to use this method, the current selection must be in a table.

Example

This example selects the second column in the first table and inserts a new column to the right of it.

```
ActiveDocument.Tables(1).Columns(2).Select  
Selection.InsertColumnsRight
```



↳ [Show All](#)

InsertCrossReference Method

Inserts a cross-reference to a heading, bookmark, footnote, or endnote, or to an item for which a caption label is defined (for example, an equation, figure, or table).

expression.**InsertCrossReference**(*ReferenceType*, *ReferenceKind*, *ReferenceItem*, *InsertAsHyperlink*, *IncludePosition*)

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

ReferenceType Required **Variant**. The type of item for which a cross-reference is to be inserted. Can be any [WdReferenceType](#) or [WdCaptionLabelID](#) constant or a user defined caption label.

WdReferenceType can be one of these WdReferenceType constants.

wdRefTypeBookmark

wdRefTypeEndnote

wdRefTypeFootnote

wdRefTypeHeading

wdRefTypeNumberedItem

WdCaptionLabelID can be one of these WdCaptionLabelID constants.

wdCaptionEquation

wdCaptionFigure

wdCaptionTable

ReferenceKind Required [WdReferenceKind](#). The information to be included in the cross-reference.

WdReferenceKind can be one of these WdReferenceKind constants.

wdContentText

wdEndnoteNumber
wdEndnoteNumberFormatted
wdEntireCaption
wdFootnoteNumber
wdFootnoteNumberFormatted
wdNumberFullContext
wdNumberNoContext
wdNumberRelativeContext
wdOnlyCaptionText
wdOnlyLabelAndNumber
wdPageNumber
wdPosition

ReferenceItem Required **Variant**. If ***ReferenceType*** is **wdRefTypeBookmark**, this argument specifies a bookmark name. For all other ***ReferenceType*** values, this argument specifies the item number or name in the **Reference type** box in the **Cross-reference** dialog box. Use the [GetCrossReferenceItems](#) method to return a list of item names that can be used with this argument.

InsertAsHyperlink Optional **Variant**. **True** to insert the cross-reference as a hyperlink to the referenced item.

IncludePosition Optional **Variant**. **True** to insert "above" or "below," depending on the location of the reference item in relation to the cross-reference.

Remarks

If you specify **wdPageNumber** for the value of *ReferenceKind*, you may need to repaginate the document in order to see the correct cross-reference information.

Example

This example inserts at the beginning of the active document a cross-reference to the page that includes the first bookmark in the document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
myBookmarks = ActiveDocument _
    .GetCrossReferenceItems(wdRefTypeBookmark)
With myRange
    .InsertBefore "Page "
    .Collapse Direction:=wdCollapseEnd
    .InsertCrossReference ReferenceType:=wdRefTypeBookmark, _
        ReferenceKind:=wdPageNumber, ReferenceItem:=myBookmarks(1)
End With
```

This example inserts a sentence that contains two cross-references: one cross-reference to heading text, and another one to the page where the heading text appears.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .InsertBefore "For more information, see "
    .Collapse Direction:=wdCollapseEnd
    .InsertCrossReference ReferenceType:=wdRefTypeHeading, _
        ReferenceKind:=wdContentText, ReferenceItem:=1
    .InsertAfter " on page "
    .Collapse Direction:=wdCollapseEnd
    .InsertCrossReference ReferenceType:=wdRefTypeHeading, _
        ReferenceKind:=wdPageNumber, ReferenceItem:=1
    .InsertAfter "."
End With
```



InsertDatabase Method

Retrieves data from a data source (for example, a separate Word document, a Microsoft Excel worksheet, or a Microsoft Access database) and inserts the data as a table in place of the specified range.

expression.InsertDatabase(Format, Style, LinkToSource, Connection, SQLStatement, SQLStatement1, PasswordDocument, PasswordTemplate, WritePasswordDocument, WritePasswordTemplate, DataSource, From, To, IncludeFields)

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

Format Optional **VARIANT**. A format listed in the **Formats** box in the **Table AutoFormat** dialog box (**Table** menu). Can be any of the [WdTableFormat](#) constants. A border is applied to the cells in the table by default.

Style Optional **VARIANT**. The attributes of the AutoFormat specified by **Format** that are applied to the table. Use the sum of any combination of the following values:

Value	Meaning
0 (zero)	None
1	Borders
2	Shading
4	Font
8	Color
16	Auto Fit
32	Heading Rows
64	Last Row
128	First Column
256	Last Column

LinkToSource Optional **Variant**. **True** to establish a link between the new table and the data source.

Connection Optional **Variant**. A range within which to perform the query specified by **SQLStatement**. How you specify the range depends on how data is retrieved. For example:

- When retrieving data through ODBC, you specify a connection string.
- When retrieving data from Microsoft Excel by using dynamic data exchange (DDE), you specify a named range or "Entire Spreadsheet."
- When retrieving data from Microsoft Access, you specify the word "Table" or "Query" followed by the name of a table or query.

SQLStatement Optional **String**. An optional query string that retrieves a subset of the data in a primary data source to be inserted into the document.

SQLStatement1 Optional **String**. If the query string is longer than 255 characters, **SQLStatement** denotes the first portion of the string and **SQLStatement1** denotes the second portion.

PasswordDocument Optional **Variant**. The password (if any) required to open the data source.

PasswordTemplate Optional **Variant**. If the data source is a Word document, this argument is the password (if any) required to open the attached template.

WritePasswordDocument Optional **Variant**. The password required to save changes to the document.

WritePasswordTemplate Optional **Variant**. The password required to save changes to the template.

DataSource Optional **Variant**. The path and file name of the data source.

From Optional **Variant**. The number of the first data record in the range of records to be inserted.

To Optional **Variant**. The number of the last data record in the range of records to be inserted.

IncludeFields Optional **Variant**. **True** to include field names from the data source in the first row of the new table.

Example

This example inserts a Microsoft Excel spreadsheet named "Data.xls" after the selection . The **Style** value (191) is a combination of the numbers 1, 2, 4, 8, 16, 32, and 128.

With Selection

```
.Collapse Direction:=wdCollapseEnd
.Range.InsertDatabase _
    Format:=wdTableFormatSimple2, Style:=191, _
    LinkToSource:=False, Connection:="Entire Spreadsheet", _
    DataSource:="C:\MSOffice\Excel\Data.xls"
```

End With



InsertDateTime Method

Inserts the current date or time, or both, either as text or as a TIME field.

expression.**InsertDateTime**(*DateTimeFormat*, *InsertAsField*,
InsertAsFullWidth, *DateLanguage*, *CalendarType*)

expression Required. An expression that returns a **Range** or **Selection** object.

DateTimeFormat Optional **Variant**. The format to be used for displaying the date or time, or both. If this argument is omitted, Microsoft Word uses the short-date style from the Windows Control Panel (**Regional Settings** icon).

InsertAsField Optional **Variant**. **True** to insert the specified information as a TIME field. The default value is **True**.

InsertAsFullWidth Optional **Variant**. **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.

DateLanguage Optional **Variant**. Sets the language in which to display the date or time. Can be either of the following **WdDateLanguage** constants: **wdDateLanguageBidi** or **wdDateLanguageLatin**. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

CalendarType Optional **Variant**. Sets the calendar type to use when displaying the date or time. Can be either of the following **WdCalendarTypeBi** constants: **wdCalendarTypeBidi** or **wdCalendarTypeGregorian**. This argument 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 inserts a TIME field for the current date. A possible result might be "November 18, 1999."

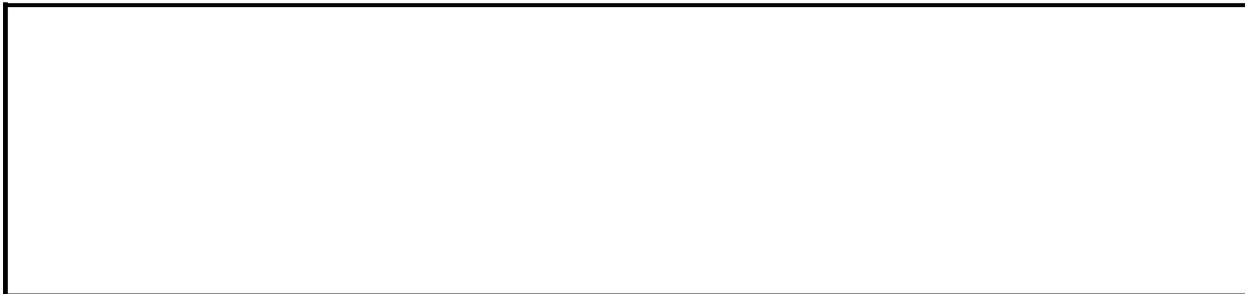
```
Selection.InsertDateTime DateTimeFormat:="MMMM dd, yyyy", _  
    InsertAsField:=True
```

This example inserts the current date at the end of the active document. A possible result might be "01/12/99."

```
With ActiveDocument.Content  
    .Collapse Direction:=wdCollapseEnd  
    .InsertDateTime DateTimeFormat:="MM/dd/yy", _  
        InsertAsField:=False  
End With
```

This example inserts a TIME field for the current date in the footer for the active document.

```
ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary).Range _  
    .InsertDateTime DateTimeFormat:="MMMM dd, yyyy", _  
    InsertAsField:=True
```



InsertFile Method

Inserts all or part of the specified file.

expression.**InsertFile**(*FileName*, *Range*, *ConfirmConversions*, *Link*, *Attachment*)

expression Required. An expression that returns a **Range** or **Selection** object.

FileName Required **String**. The path and file name of the file to be inserted. If you don't specify a path, Word assumes the file is in the current folder.

Range Optional **Variant**. If the specified file is a Word document, this parameter refers to a bookmark. If the file is another type (for example, a Microsoft Excel worksheet), this parameter refers to a named range or a cell range (for example, R1C1:R3C4).

ConfirmConversions Optional **Variant**. **True** to have Word prompt you to confirm conversion when inserting files in formats other than the Word Document format.

Link Optional **Variant**. **True** to insert the file by using an INCLUDETEXT field.

Attachment Optional **Variant**. **True** to insert the file as an attachment to an e-mail message.

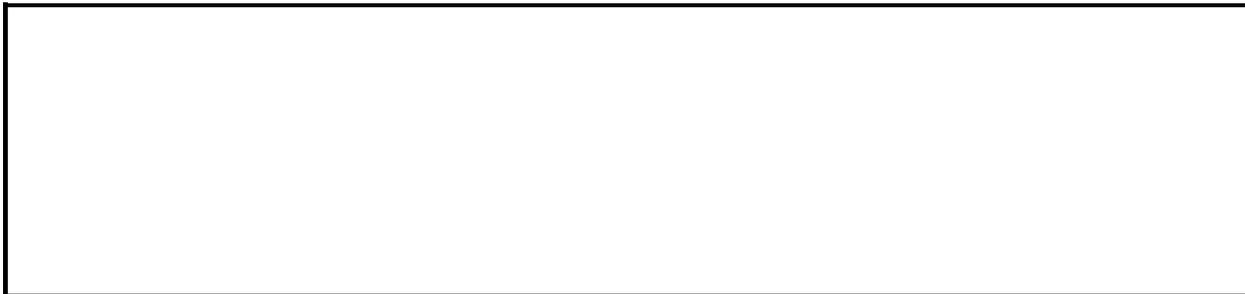
Example

This example uses an INCLUDETEXT field to insert the TEST.DOC file at the insertion point.

```
Selection.Collapse Direction:=wdCollapseEnd
Selection.InsertFile FileName:="C:\TEST.DOC", Link:=True
```

This example creates a new document and then inserts the contents of each text file in the C:\TMP folder into the new document.

```
Documents.Add
ChDir "C:\TMP"
myName = Dir("*.TXT")
While myName <> ""
    With Selection
        .InsertFile FileName:=myName, ConfirmConversions:=False
        .InsertParagraphAfter
        .InsertBreak Type:=wdSectionBreakNextPage
        .Collapse Direction:=wdCollapseEnd
    End With
    myName = Dir()
Wend
```



InsertFormula Method

Inserts an = (Formula) field that contains a formula at the selection.

Note The formula replaces the selection, if the selection isn't collapsed.

expression.**Formula**(*Formula*, *NumberFormat*)

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

Formula Optional **Variant**. The mathematical formula you want the = (Formula) field to evaluate. Spreadsheet-type references to table cells are valid. For example, "=SUM(A4:C4)" specifies the first three values in the fourth row. For more information about the = (Formula) field, see [Field codes:= \(Formula\) field](#).

NumberFormat Optional **Variant**. A format for the result of the = (Formula) field. For information about the types of formats you can apply, see [Numeric Picture \(\#\) field switch](#).

Remarks

If you're using a spreadsheet application, such as Microsoft Excel, embedding all or part of a worksheet in a document is often easier than using the = (Formula) field in a table.

The **Formula** argument is optional only if the selection is in a cell and there's at least one cell that contains a value above or to the left of the cell that contains the insertion point. If the cells above the insertion point contain values, the inserted field is {=SUM(ABOVE)}; if the cells to the left of the insertion point contain values, the inserted field is {=SUM(LEFT)}. If both the cells above the insertion point and the cells to the left of it contain values, Microsoft Word uses the following rules to determine which SUM function to insert:

- If the cell immediately above the insertion point contains a value, Word inserts {=SUM(ABOVE)}.
- If the cell immediately above the insertion point doesn't contain a value but the cell immediately to the left of the insertion point does, Word inserts {=SUM(LEFT)}.
- If neither cell immediately above the insertion point nor the cell immediately below it contains a value, Word inserts {=SUM(ABOVE)}.
- If you don't specify **Formula** and all the cells above and to the left of the insertion point are empty, using the = (Formula) field causes an error.

Example

This example creates a table with three rows and three columns at the beginning of the active document and then calculates the average of all the numbers in the first column.

```
Set MyRange = ActiveDocument.Range(0, 0)
Set myTable = ActiveDocument.Tables.Add(MyRange, 3, 3)
With myTable
    .Cell(1, 1).Range.InsertAfter "100"
    .Cell(2, 1).Range.InsertAfter "50"
    .Cell(3, 1).Select
End With
Selection.InsertFormula Formula:="=Average(Above)"
```

The example inserts a formula field that's subtracted from a value represented by the bookmark named "GrossSales." The result is formatted with a dollar sign.

```
Selection.Collapse Direction:=wdCollapseStart
Selection.InsertFormula Formula:= "=GrossSales-45,000.00", _
    NumberFormat:="$#,##0.00"
```



InsertParagraph Method

-
Replaces the specified range or selection with a new paragraph.

Note After this method has been used, the range or selection is the new paragraph.

expression.**InsertParagraph**

expression Required. An expression that returns a **Range** or **Selection** object.

Remarks

If you don't want to replace the range or selection, use the **Collapse** method before using this method. The **InsertParagraphAfter** method inserts a new paragraph following a **Range** or **Selection** object.

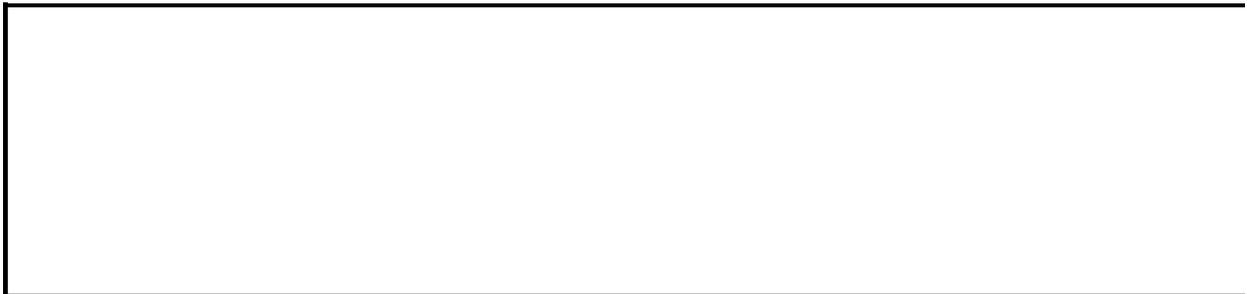
Example

This example inserts a new paragraph at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(0, 0)
With myRange
    .InsertParagraph
    .InsertBefore "Dear Sirs,"
End With
```

This example collapses the selection and then inserts a paragraph mark at the insertion point.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .InsertParagraph
    .Collapse Direction:=wdCollapseEnd
End With
```



InsertParagraphAfter Method

-
Inserts a paragraph mark after a range or selection.

Note After this method is applied, the range or selection expands to include the new paragraph.

expression.**InsertParagraphAfter**

expression Required. An expression that returns a **Range** or **Selection** object.

Example

This example inserts a new paragraph after the current paragraph.

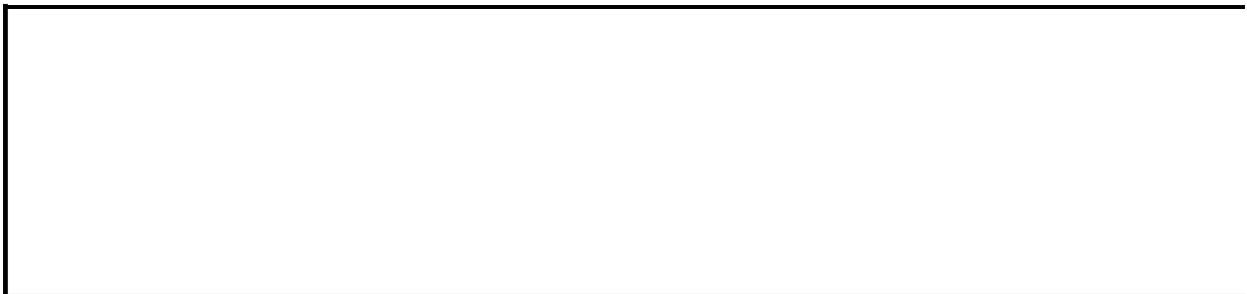
```
With Selection
    .Move Unit:=wdParagraph
    .InsertParagraphAfter
    .Collapse Direction:=wdCollapseStart
End With
```

This example inserts text as a new paragraph at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(0, 0)
With myRange
    .InsertBefore "Title"
    .ParagraphFormat.Alignment = wdAlignParagraphCenter
    .InsertParagraphAfter
End With
```

This example inserts a paragraph at the end of the active document. The **Content** property returns a **Range** object.

```
ActiveDocument.Content.InsertParagraphAfter
```



InsertParagraphBefore Method

-
Inserts a new paragraph before the specified selection or range.

Note After this method is applied, the range or selection expands to include the new paragraph.

expression.**InsertParagraphBefore**

expression Required. An expression that returns a **Selection** or **Range** object.

Example

This example inserts a new paragraph at the beginning of the active document.

```
ActiveDocument.Range(Start:=0, End:=0).InsertParagraphBefore
```

This example inserts the text "Hello" as a new paragraph before the selection.

```
With Selection  
    .InsertParagraphBefore  
    .InsertBefore "Hello"  
End With
```



InsertRows Method

-

Inserts the specified number of new rows above the row that contains the selection. If the selection isn't in a table, an error occurs.

Note You can also insert rows by using the [Add](#) method of the **Rows** object.

expression.**InsertRows**(*NumRows*)

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

NumRows Optional **Variant**. The number of rows to be added.

Example

This example inserts two new rows above the row that contains the selection, and then it removes the borders from the new rows.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.InsertRows NumRows:=2
    Selection.Borders.Enable =False
End If
```



InsertRowsAbove Method

-
Inserts rows above the current selection.

expression.**InsertRowsAbove**

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

Remarks

Microsoft Word inserts as many rows as there are in the current selection.

In order to use this method, the current selection must be in a table.

Example

This example selects the second row in the first table and inserts a new row above it.

```
ActiveDocument.Tables(1).Rows(2).Select  
Selection.InsertRowsAbove
```



InsertRowsBelow Method

-
Inserts rows below the current selection.

expression.**InsertRowsBelow**

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

Remarks

Microsoft Word inserts as many rows as there are in the current selection.

In order to use this method, the current selection must be in a table.

Example

This example selects the second row in the first table and inserts a new row below it.

```
ActiveDocument.Tables(1).Rows(2).Select  
Selection.InsertRowsBelow
```



InsertStyleSeparator Method

-

Inserts a special hidden paragraph mark that allows Microsoft Word to join paragraphs formatted using different paragraph styles, so lead-in headings can be inserted into a table of contents.

expression.**InsertStyleSeparator**

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

Example

This example inserts a style separator after every paragraph formatted with the built-in "Heading 4" style.

Note The paragraph count is inside the **Do...Loop** because when Word inserts the style separator, the two paragraphs become one paragraph, so the paragraph count for the document changes as the procedure runs.

```
Sub InlineHeading()  
    Dim intCount As Integer  
    Dim intParaCount As Integer  
  
    intCount = 1  
  
    With ThisDocument  
        Do  
            'Look for all paragraphs formatted with "Heading 4" styl  
            If .Paragraphs(Index:=intCount).Style = "Heading 4" Then  
                .Paragraphs(Index:=intCount).Range.Select  
  
                'Insert a style separator if paragraph  
                'is formatted with a "Heading 4" style  
                Selection.InsertStyleSeparator  
            End If  
            intCount = intCount + 1  
            intParaCount = .Paragraphs.Count  
        Loop Until intCount = intParaCount  
    End With  
End Sub
```



InsertSymbol Method

Inserts a symbol in place of the specified range or selection.

Note If you don't want to replace the range or selection, use the **Collapse** method before you use this method.

expression.**InsertSymbol**(*CharacterNumber*, *Font*, *Unicode*, *Bias*)

expression Required. An expression that returns a **Range** or **Selection** object.

CharacterNumber Required **Long**. The character number for the specified symbol. This value will always be the sum of 31 and the number that corresponds to the position of the symbol in the table of symbols (counting from left to right). For example, to specify a delta character at position 37 in the table of symbols in the Symbol font, set **CharacterNumber** to 68.

Font Optional **Variant**. The name of the font that contains the symbol.

Unicode Optional **Variant**. **True** to insert the unicode character specified by **CharacterNumber**; **False** to insert the ANSI character specified by **CharacterNumber**. The default value is **False**.

Bias Optional **Variant**. Sets the font bias for symbols. This argument is useful for setting the correct font bias for East Asian characters. Can be one of the following **WdFontBias** constants: **wdFontBiasDefault**, **wdFontBiasDontCare**, or **wdFontBiasFareast**. This argument 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 inserts a double-headed arrow at the insertion point.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .InsertSymbol CharacterNumber:=171, _
        Font:="Symbol", Unicode:=False
End With
```

This example inserts a bullet and a tab stop at the beginning of the first paragraph in the selection.

```
Set myRange = Selection.Paragraphs(1).Range
With myRange
    .Collapse Direction:=wdCollapseStart
    .InsertSymbol CharacterNumber:=183, _
        Font:="Symbol", Unicode:=False
    .MoveStart Unit:=wdCharacter, Count:=1
    .InsertAfter vbTab
End With
```



↳ [Show All](#)

InStory Method

-
True if the selection or range to which this method is applied is in the same [story](#) as the range specified by the **Range** argument.

Note A range can belong to only one story.

expression.**InStory**(**Range**)

expression Required. An expression that returns a **Range** or **Selection** object.

Range Required **Range** object. The **Range** object whose story is compared with the story that contains *expression*.

Example

This example determines whether the selection is in the same story as the first paragraph in the active document. The message box displays "False" because the selection is in the primary header story and the first paragraph is in the main text story.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .SeekView = wdSeekCurrentPageHeader
End With
same = Selection.InStory(ActiveDocument.Paragraphs(1).Range)
MsgBox same
```

This example determines whether Range1 and Range2 are in the same story. If they are, bold formatting is applied to Range1.

```
Set Range1 = Selection.Words(1)
Set Range2 = ActiveDocument.Range(Start:=20, End:=100)
If Range1.InStory(Range:=Range2) = True Then
    Range1.Font.Bold = True
End If
```



IsEqual Method

True if the selection or range to which this method is applied is equal to the range specified by the **Range** argument. This method compares the starting and ending character positions, as well as the story type. If all three of these items are the same for both objects, the objects are equal.

expression.**IsEqual**(**Range**)

expression Required. An expression that returns a **Range** or **Selection** object.

Range Required **Range** object. The **Range** object that's compared with *expression*.

Example

This example compares the selection with the second paragraph in the active document. If the selection isn't equal to the second paragraph, the second paragraph is selected.

```
If Selection.IsEqual(ActiveDocument _  
    .Paragraphs(2).Range) = False Then  
    ActiveDocument.Paragraphs(2).Range.Select  
End If
```

This example compares Range1 with Range2 to determine whether they're equal. If the two ranges are equal, the content of Range1 is deleted.

```
Set Range1 = Selection.Words(1)  
Set Range2 = ActiveDocument.Words(3)  
If Range1.IsEqual(Range:=Range2) = True Then  
    Range1.Delete  
End If
```



↳ [Show All](#)

ItalicRun Method

-

Adds the italic character format to or removes it from the current [run](#). If the run contains a mix of italic and non-italic text, this method adds the italic character format to the entire run.

expression.**ItalicRun**

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

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example toggles the italic formatting for the current selection.

Selection.**ItalicRun**



↳ [Show All](#)

Item Method

▶ [Item method as it applies to the **Borders** object.](#)

Returns a border in a range or selection.

expression.**Item**(*Index*)

expression Required. An expression that returns a [Borders](#) object.

Index Required [WdBorderType](#). The border to be returned.

WdBorderType can be one of these WdBorderType constants.

wdBorderBottom

wdBorderDiagonalDown

wdBorderDiagonalUp

wdBorderHorizontal

wdBorderLeft

wdBorderRight

wdBorderTop

wdBorderVertical

▶ [Item method as it applies to the **Dialogs** object.](#)

Returns a dialog in Microsoft Word.

expression.**Item**(*Index*)

expression Required. An expression that returns a [Dialogs](#) object.

Index Required [WdWordDialog](#). A constant that specifies the dialog.

WdWordDialog can be one of these WdWordDialog constants.

wdDialogMailMergeInsertSet
wdDialogConnect
wdDialogConsistencyChecker
wdDialogControlRun
wdDialogConvertObject
wdDialogCopyFile
wdDialogCreateAutoText
wdDialogDocumentStatistics
wdDialogDrawAlign
wdDialogDrawSnapToGrid
wdDialogEditAutoText
wdDialogEditCreatePublisher
wdDialogEditFind
wdDialogEditFrame
wdDialogEditGoTo
wdDialogEditGoToOld
wdDialogEditLinks
wdDialogEditObject
wdDialogEditPasteSpecial
wdDialogEditPublishOptions
wdDialogEditReplace
wdDialogEditStyle
wdDialogEditSubscribeOptions
wdDialogEditSubscribeTo
wdDialogEditTOACategory
wdDialogEmailOptions
wdDialogFileDocumentLayout
wdDialogFileFind
wdDialogFileMacCustomPageSetupGX
wdDialogFileMacPageSetup
wdDialogFileMacPageSetupGX

wdDialogFileNew
wdDialogFileOpen
wdDialogFilePageSetup
wdDialogFilePrint
wdDialogFilePrintOneCopy
wdDialogFilePrintSetup
wdDialogFileRoutingSlip
wdDialogFileSaveAs
wdDialogFileSaveVersion
wdDialogFileSummaryInfo
wdDialogFileVersions
wdDialogFitText
wdDialogFontSubstitution
wdDialogFormatAddrFonts
wdDialogFormatBordersAndShading
wdDialogFormatBulletsAndNumbering
wdDialogFormatCallout
wdDialogFormatChangeCase
wdDialogFormatColumns
wdDialogFormatDefineStyleBorders
wdDialogFormatDefineStyleFont
wdDialogFormatDefineStyleFrame
wdDialogFormatDefineStyleLang
wdDialogFormatDefineStylePara
wdDialogFormatDefineStyleTabs
wdDialogFormatDrawingObject
wdDialogFormatDropCap
wdDialogFormatEncloseCharacters
wdDialogFormatFont
wdDialogFormatFrame
wdDialogFormatPageNumber

wdDialogFormatParagraph
wdDialogFormatPicture
wdDialogFormatRetAddrFonts
wdDialogFormatSectionLayout
wdDialogFormatStyle
wdDialogFormatStyleGallery
wdDialogFormatStylesCustom
wdDialogFormatTabs
wdDialogFormatTheme
wdDialogFormFieldHelp
wdDialogFormFieldOptions
wdDialogFrameSetProperties
wdDialogHelpAbout
wdDialogHelpWordPerfectHelp
wdDialogHelpWordPerfectHelpOptions
wdDialogHorizontalInVertical
wdDialogIMESetDefault
wdDialogInsertAddCaption
wdDialogInsertAutoCaption
wdDialogInsertBookmark
wdDialogInsertBreak
wdDialogInsertCaption
wdDialogInsertCaptionNumbering
wdDialogInsertCrossReference
wdDialogInsertDatabase
wdDialogInsertDateTime
wdDialogInsertField
wdDialogInsertFile
wdDialogInsertFootnote
wdDialogInsertFormField
wdDialogInsertHyperlink

wdDialogInsertIndex
wdDialogInsertIndexAndTables
wdDialogInsertMergeField
wdDialogInsertNumber
wdDialogInsertObject
wdDialogInsertPageNumbers
wdDialogInsertPicture
wdDialogInsertSubdocument
wdDialogInsertSymbol
wdDialogInsertTableOfAuthorities
wdDialogInsertTableOfContents
wdDialogInsertTableOfFigures
wdDialogLetterWizard
wdDialogListCommands
wdDialogMailMerge
wdDialogMailMergeCheck
wdDialogMailMergeCreateDataSource
wdDialogMailMergeCreateHeaderSource
wdDialogMailMergeFieldMapping
wdDialogMailMergeFindRecord
wdDialogMailMergeHelper
wdDialogMailMergeInsertAddressBlock
wdDialogMailMergeInsertAsk
wdDialogMailMergeInsertFields
wdDialogMailMergeInsertFillIn
wdDialogMailMergeInsertGreetingLine
wdDialogMailMergeInsertIf
wdDialogMailMergeInsertNextIf
wdDialogMailMergeInsertSkipIf
wdDialogMailMergeOpenDataSource
wdDialogMailMergeOpenHeaderSource

wdDialogMailMergeQueryOptions
wdDialogMailMergeRecipients
wdDialogMailMergeUseAddressBook
wdDialogMarkCitation
wdDialogMarkIndexEntry
wdDialogMarkTableOfContentsEntry
wdDialogNewToolbar
wdDialogNoteOptions
wdDialogOrganizer
wdDialogPhoneticGuide
wdDialogReviewAfmtRevisions
wdDialogSearch
wdDialogTableAutoFormat
wdDialogTableCellOptions
wdDialogTableColumnWidth
wdDialogTableDeleteCells
wdDialogTableFormatCell
wdDialogTableFormula
wdDialogTableInsertCells
wdDialogTableInsertRow
wdDialogTableInsertTable
wdDialogTableOfCaptionsOptions
wdDialogTableOfContentsOptions
wdDialogTableProperties
wdDialogTableRowHeight
wdDialogTableSort
wdDialogTableSplitCells
wdDialogTableTableOptions
wdDialogTableToText
wdDialogTableWrapping
wdDialogTCSCTranslator

wdDialogTextToTable
wdDialogToolsAcceptRejectChanges
wdDialogToolsAdvancedSettings
wdDialogToolsAutoCorrect
wdDialogToolsAutoCorrectExceptions
wdDialogToolsAutoManager
wdDialogToolsAutoSummarize
wdDialogToolsBulletsNumbers
wdDialogToolsCompareDocuments
wdDialogToolsCreateDirectory
wdDialogToolsCreateEnvelope
wdDialogToolsCreateLabels
wdDialogToolsCustomize
wdDialogToolsCustomizeKeyboard
wdDialogToolsCustomizeMenuBar
wdDialogToolsCustomizeMenus
wdDialogToolsDictionary
wdDialogToolsEnvelopesAndLabels
wdDialogToolsHangulHanjaConversion
wdDialogToolsHighlightChanges
wdDialogToolsHyphenation
wdDialogToolsLanguage
wdDialogToolsMacro
wdDialogToolsMacroRecord
wdDialogToolsManageFields
wdDialogToolsMergeDocuments
wdDialogToolsOptions
wdDialogToolsOptionsAutoFormat
wdDialogToolsOptionsAutoFormatAsYouType
wdDialogToolsOptionsBidi
wdDialogToolsOptionsCompatibility

wdDialogToolsOptionsEdit
wdDialogToolsOptionsFileLocations
wdDialogToolsOptionsFuzzy
wdDialogToolsOptionsGeneral
wdDialogToolsOptionsPrint
wdDialogToolsOptionsSave
wdDialogToolsOptionsSpellingAndGrammar
wdDialogToolsOptionsTrackChanges
wdDialogToolsOptionsTypography
wdDialogToolsOptionsUserInfo
wdDialogToolsOptionsView
wdDialogToolsProtectDocument
wdDialogToolsProtectSection
wdDialogToolsRevisions
wdDialogToolsSpellingAndGrammar
wdDialogToolsTemplates
wdDialogToolsThesaurus
wdDialogToolsUnprotectDocument
wdDialogToolsWordCount
wdDialogTwoLinesInOne
wdDialogUpdateTOC
wdDialogViewZoom
wdDialogWebOptions
wdDialogWindowActivate

► [Item method as it applies to the **HeadersFooters** object.](#)

Returns a header or footer in a range or selection.

expression.**Item**(*Index*)

expression Required. An expression that returns a [HeadersFooters](#) object.

Index Required [WdHeaderFooterIndex](#). A constant that specifies the header or footer in the selection.

WdHeaderFooterIndex can be one of these WdHeaderFooterIndex constants.

wdHeaderFooterEvenPages

wdHeaderFooterFirstPage

wdHeaderFooterPrimary

▶ [Item method as it applies to the ListGalleries object.](#)

Returns the type of list (bulleted, numbered or outline) from the list template gallery.

expression.**Item**(*Index*)

expression Required. An expression that returns a [ListGalleries](#) object.

Index Required [WdListGalleryType](#). A constant that specifies the type of list.

WdListGalleryType can be one of these WdListGalleryType constants.

wdBulletGallery

wdNumberGallery

wdOutlineNumberGallery

▶ [Item method as it applies to the MappedDataFields object.](#)

Returns a specified mapped data field.

expression.**Item**(*Index*)

expression Required. An expression that returns a [MappedDataFields](#) object.

Index Required [WdMappedDataFields](#). The specified mapped data field.

WdMappedDataFields can be one of these WdMappedDataFields constants.

wdAddress1

wdAddress2

wdBusinessFax
wdBusinessPhone
wdCity
wdCompany
wdCountryRegion
wdCoutesyTitle
wdEmailAddress
wdFirstName
wdHomeFax
wdHomePhone
wdJobTitle
wdLastName
wdMiddleName
wdNickname
wdPostalCode
wdSpouseCourtesyTitle
wdSpouseFirstName
wdSpouseLastName
wdSpouseMiddleName
wdSpouseNickname
wdState
wdSuffix
wdUniqueIdentifier
wdWebPageURL

► [Item method as it applies to the **StoryRanges** object.](#)

Returns a single story of a range or selection as a [Range](#) object.

expression.**Item**(*Index*)

expression Required. An expression that returns a [StoryRanges](#) object.

Index Required [WdStoryType](#). The specified story type.

WdStoryType can be one of these WdStoryType constants.

wdCommentsStory

wdEndnotesStory

wdEvenPagesFooterStory

wdEvenPagesHeaderStory

wdFirstPageFooterStory

wdFirstPageHeaderStory

wdFootnotesStory

wdMainTextStory

wdPrimaryFooterStory

wdPrimaryHeaderStory

wdTextFrameStory

▶ [Item method as it applies to the TaskPanels object.](#)

Returns the specified task pane as a [TaskPane](#) object.

expression.**Item**(*Index*)

expression Required. An expression that returns a [TaskPanels](#) object.

Index Required [WdTaskPanels](#). The specified task pane.

WdTaskPanels can be one of these WdTaskPanels constants.

wdTaskPaneFormatting

wdTaskPaneInspector

wdTaskPaneMailMerge

wdTaskPaneSearch

wdTaskPaneTranslate

▶ [Item method as it applies to the Zooms object.](#)

Returns the specified [Zoom](#) object.

expression.**Item**(*Index*)

expression Required. An expression that returns a [Zooms](#) object.

Index Required [WdViewType](#). The specified zoom type.

WdViewType can be one of these WdViewType constants.

wdMasterView

wdNormalView

wdOutlineView

wdPrintPreview

wdPrintView

wdWebView

▶ [Item method as it applies to all other objects in the Applies To list.](#)

Returns an individual object in a collection.

expression.**Item**(*Index*)

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

Index Required **Variant** or **Long**. The individual object to be returned.

For the following [objects](#), *Index* can be a **Variant** indicating the ordinal position or a string representing the name of the individual object.

AddIns

AutoCaptions

AutoCorrectEntries

AutoTextEntries

Bookmarks

CanvasShapes

CaptionLabels

CustomLabels

CustomProperties

DiagramNodeChildren
DiagramNodes
Dictionaries
Documents
EmailSignatureEntries
FileConverters
FirstLetterExceptions
FormFields
GroupShapes
HangulAndAlphabetExceptions
HangulHanjaConversionDictionaries
Hyperlinks
Languages
ListEntries
ListTemplates
MailMergeDataFields
MailMergeFieldNames
OtherCorrectionsExceptions
ReadabilityStatistics
Reviewers
ShapeRange
Shapes
ShapeNodes
SmartTags
Styles
StyleSheets
TablesOfAuthoritiesCategories
TabStops
Tasks
Templates
TwoInitialCapsExceptions
Variables
Windows

For the following [objects](#), *Index* can be a **Long** indicating the ordinal position of the individual object.

Adjustments

Cells

Characters

Columns

Comments

Endnotes

Fields

FontNames

Footnotes

Frames

HeadingStyles

HTMLDivisions

Indexes

InlineShapes

KeyBindings

KeysBoundTo

Lists

ListLevels

ListParagraphs

MailMergeFields

PageNumbers

Panes

Paragraphs

ProofreadingErrors

RecentFiles

Revisions

Rows

Sections

Sentences

SpellingSuggestions

Subdocuments

Tables

TablesOfAuthorities

TablesOfContents

TablesOfFigures

TextColumns

Versions

Words

Example

▶ [As it applies to the **Bookmarks** object.](#)

This example selects the bookmark named "temp" in the active document.

```
Sub BookmarkItem()  
    If ActiveDocument.Bookmarks.Exists("temp") = True Then  
        ActiveDocument.Bookmarks.Item("temp").Select  
    End If  
End Sub
```

▶ [As it applies to the **Borders** object.](#)

This example inserts a double border above the first paragraph in the active document.

```
Sub BorderItem()  
    ActiveDocument.Paragraphs(1).Borders.Item(wdBorderTop) _  
        .LineStyle = wdLineStyleDouble  
End Sub
```

▶ [As it applies to the **Dialogs** object.](#)

This example displays the Page Setup dialog.

```
Sub DialogItem()  
    Application.Dialogs.Item(wdDialogFileDocumentLayout).Display  
End Sub
```

▶ [As it applies to the **Documents** object.](#)

This example displays the name of the first document in the **Documents** collection.

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

▶ [As it applies to the **HeadersFooters** object.](#)

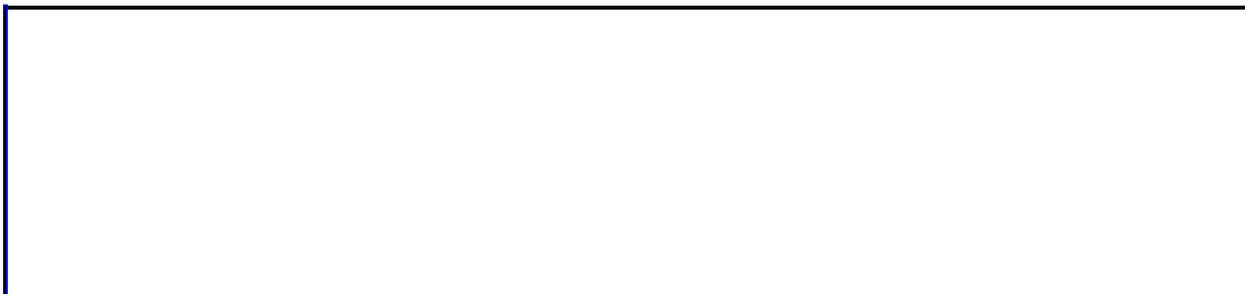
This example creates and formats a first page header in the active document.

```
Sub HeadFootItem()  
    ActiveDocument.PageSetup.DifferentFirstPageHeaderFooter = True  
    With ActiveDocument.Sections(1).Headers _  
        .Item(wdHeaderFooterFirstPage).Range  
        .InsertBefore "Sales Report"  
        With .Font  
            .Bold = True  
            .Size = "15"  
            .Color = wdColorBlue  
        End With  
        .Paragraphs.Alignment = wdAlignParagraphCenter  
    End With  
End Sub
```

► [As it applies to the **ListEntries** object.](#)

This example clears all the items from the drop-down form field named "Colors" and then adds two color names. The **Item** method is used to display the first color in the drop-down form field.

```
Sub ListEntryItem()  
    Dim d As DropDown  
    Set d = ActiveDocument.FormFields.Add _  
        (Range:=Selection.Range, _  
        Type:=wdFieldFormDropDown).DropDown  
    With d.ListEntries  
        .Add Name:="Black"  
        .Add Name:="Green"  
    End With  
    MsgBox d.ListEntries.Item(1).Name  
End Sub
```



Key Method

Returns a [KeyBinding](#) object that represents the specified custom key combination. If the key combination doesn't exist, this method returns **Nothing**.

expression.**Key**(*KeyCode*, *KeyCode2*)

expression Required. An expression that returns a **KeyBindings** or **KeysBoundTo** object.

KeyCode Required **Long**. A key you specify by using one of the **WdKey** constants.

KeyCode2 Optional **Variant**. A second key you specify by using one of the **WdKey** constants.

Remarks

You can use the [BuildKeyCode](#) method to create the *KeyCode* or *KeyCode2* argument.

Example

This example assigns the ALT+F4 key combination to the Arial font and then displays the number of items in the **KeyBindings** collection. The example then clears the key combinations (returns it to its default setting) and redisplay the number of items in the **KeyBindings** collection.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyAlt, wdKeyF4), _
    KeyCategory:=wdKeyCategoryFont, Command:="Arial"
MsgBox KeyBindings.Count & " keys in KeyBindings collection"
KeyBindings.Key(KeyCode:=BuildKeyCode(wdKeyAlt, wdKeyF4)).Clear
MsgBox KeyBindings.Count & " keys in KeyBindings collection"
```

This example assigns the CTRL+SHIFT+U key combination to the macro named "Macro1" in the active document. The example uses the **Key** property to return a **KeyBinding** object so that Word can retrieve and display the command name.

```
CustomizationContext = ActiveDocument
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyControl, _
    wdKeyShift, wdKeyU), KeyCategory:=wdKeyCategoryMacro, _
    Command:="Macro1"
MsgBox KeyBindings.Key(BuildKeyCode(wdKeyControl, _
    wdKeyShift, wdKeyU)).Command
```

This example determines whether the CTRL+SHIFT+A key combination is part of the **KeyBindings** collection.

```
Dim kbTemp As KeyBinding

CustomizationContext = NormalTemplate
Set kbTemp = KeyBindings.Key(BuildKeyCode(wdKeyControl, _
    wdKeyShift, wdKeyA))
If (kbTemp Is Nothing) Then MsgBox _
    "Key is not in the KeyBindings collection"
```



Keyboard Method

Returns or sets the keyboard language and layout settings.

expression.**Keyboard**(*LangId*)

expression Required. An expression that returns an **Application** object.

LangId Optional **Long**. The language and layout combination to which Microsoft Word sets the keyboard. If this argument is omitted, the method returns the current language and layout setting.

Remarks

Microsoft Windows tracks keyboard language and layout settings using a variable type called an input language handle, often referred to as an HKL. The low word of the handle is a language ID, and the high word is a handle to a keyboard layout.

Example

This example assigns the current keyboard language and layout setting to a variable.

```
Dim lngKeyboard As Long
```

```
lngKeyboard = Application.Keyboard
```



KeyboardBidi Method

-
Sets the keyboard language to a right-to-left language and the text entry direction to right-to-left.

expression.**KeyboardBidi**

expression Required. An expression that returns an **Application** object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example configures the keyboard for right-to-left language entry.

`Application.KeyboardBidi`



KeyboardLatin Method

-
Sets the keyboard language to a left-to-right language and the text entry direction to left-to-right.

expression.**KeyboardLatin**

expression Required. An expression that returns an **Application** object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example configures the keyboard for left-to-right language entry.

`Application.KeyboardLatin`



KeyString Method

Returns the key combination string for the specified keys (for example, CTRL+SHIFT+A).

expression.**KeyString**(*KeyCode*, *KeyCode2*)

expression Optional. An expression that returns an **Application** object.

KeyCode Required **Long**. A key you specify by using one of the **WdKey** constants.

KeyCode2 Optional **Variant**. A second key you specify by using one of the **WdKey** constants.

Remarks

You can use the [BuildKeyCode](#) method to create the *KeyCode* or *KeyCode2* argument.

Example

This example displays the key combination string (CTRL+SHIFT+A) for the following **WdKey** constants: **wdKeyControl**, **wdKeyShift**, and **wdKeyA**.

```
CustomizationContext = ActiveDocument.AttachedTemplate  
MsgBox KeyString(KeyCode:=BuildKeyCode(wdKeyControl, _  
    wdKeyShift, wdKeyA))
```



LabelOptions Method

-
Displays the **Label Options** dialog box.

expression.**LabelOptions**

expression Required. An expression that returns a [MailingLabel](#) object.

Remarks

The **LabelOptions** method works only if the document is the main document of a mailing labels mail merge.

Example

This example determines if the current document is a Mailing Label document and, if it is, displays the **Label Options** dialog box.

```
Sub LabelOps()  
    If ThisDocument.MailMerge _  
        .MainDocumentType = wdMailingLabels Then  
        Application.MailingLabel.LabelOptions  
    End If  
End Sub
```



LargeScroll Method

-

Scrolls a window or pane by the specified number of screens. This method is equivalent to clicking just before or just after the scroll boxes on the horizontal and vertical scroll bars.

expression.**LargeScroll**(*Down, Up, ToRight,ToLeft*)

expression Required. An expression that returns a **Pane** or **Window** object.

Down Optional **Variant**. The number of screens to scroll the window down.

Up Optional **Variant**. The number of screens to scroll the window up.

ToRight Optional **Variant**. The number of screens to scroll the window to the right.

ToLeft Optional **Variant**. The number of screens to scroll the window to the left.

Remarks

If ***Down*** and ***Up*** are both specified, the window is scrolled by the difference of the arguments. For example, if ***Down*** is 2 and ***Up*** is 4, the window is scrolled up two screens. Similarly, if ***ToLeft*** and ***ToRight*** are both specified, the window is scrolled by the difference of the arguments.

Any of these arguments can be a negative number. If no arguments are specified, the window is scrolled down one screen.

Example

This example scrolls the active window down one screen.

```
ActiveDocument.ActiveWindow.LargeScroll Down:=1
```

This example splits the active window and then scrolls up two screens and to the right one screen.

```
With ActiveDocument.ActiveWindow  
    .Split = True  
    .LargeScroll Up:=2, ToRight:=1  
End With
```



LinesToPoints Method

-
Converts a measurement from lines to points (1 line = 12 points). Returns the converted measurement as a **Single**.

expression.**LinesToPoints**(*Lines*)

expression Optional. An expression that returns an **Application** object.

Lines Required **Single**. The line value to be converted to points.

Example

This example sets the paragraph line spacing in the selection to three lines.

```
With Selection.ParagraphFormat
    .LineSpacingRule = wdLineSpaceMultiple
    .LineSpacing = LinesToPoints(3)
End With
```



LinkToListTemplate Method

-

Links the specified style to a list template so that the style's formatting can be applied to lists.

expression.**LinkToListTemplate**(*ListTemplate*, *ListLevelNumber*)

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

ListTemplate Required **ListTemplate** object. The list template that the style is to be linked to.

ListLevelNumber Optional **Variant**. An integer corresponding to the list level that the style is to be linked to. If this argument is omitted, then the level of the style is used.

Example

This example creates a new list template and then links heading styles 1 through 9 to levels 1 through 9. The new list template is then applied to the document. Any paragraphs formatted as heading styles will assume the numbering from the list template.

```
Dim ltTemp As ListTemplate
Dim intLoop As Integer

Set ltTemp = _
    ActiveDocument.ListTemplates.Add(OutlineNumbered:=True)

For intLoop = 1 To 9
    With ltTemp.ListLevels(intLoop)
        .NumberStyle = wdListNumberStyleArabic
        .NumberPosition = InchesToPoints(0.25 * (intLoop - 1))
        .TextPosition = InchesToPoints(0.25 * intLoop)
        .NumberFormat = "%" & intLoop & "."
    End With
    With ActiveDocument.Styles("Heading " & intLoop)
        .LinkToListTemplate ListTemplate:=ltTemp
    End With
Next intLoop

ActiveDocument.Content.ListFormat.ApplyListTemplate _
    ListTemplate:=ltTemp
```



ListCommands Method

-
Creates a new document and then inserts a table of Word commands along with their associated shortcut keys and menu assignments.

expression.**ListCommands**(*ListAllCommands*)

expression Required. An expression that returns an **Application** object.

ListAllCommands Required **Boolean**. **True** to include all Word commands and their assignments (whether customized or built-in). **False** to include only commands with customized assignments.

Example

This example creates a new document that lists all Word commands along with their associated shortcut keys and menu assignments. The example then prints and closes the new document without saving changes.

```
Application.ListCommands ListAllCommands:=True  
With ActiveDocument  
    .PrintOut  
    .Close SaveChanges:=wdDoNotSaveChanges  
End With
```



ListIndent Method

-
Increases the list level of the paragraphs in the range for the specified **ListFormat** object, in increments of one level.

expression.**ListIndent**

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

Example

This example indents each paragraph in the first list in document one by one level.

```
Documents(1).Lists(1).Range.ListFormat.ListIndent
```

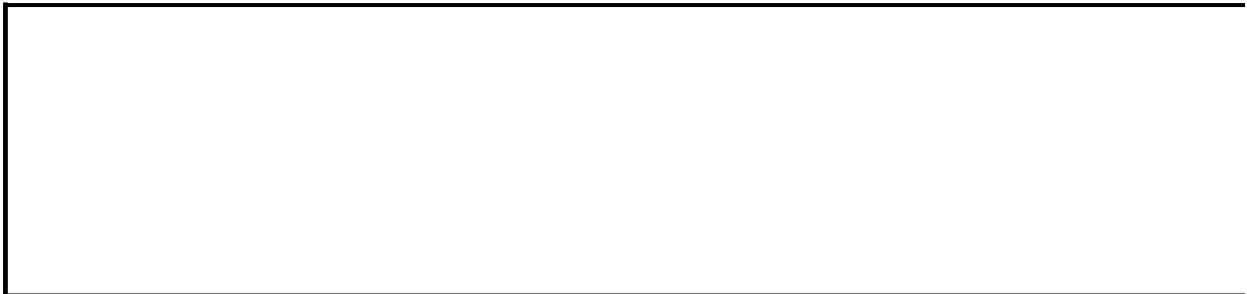
This example formats paragraphs four through eight in the active document as an outline-numbered list, and then it indents the paragraphs one level.

```
Dim docActive As Document
Dim rngTemp As Range

Set docActive = ActiveDocument

Set rngTemp = _
    docActive.Range( _
        Start:=docActive.Paragraphs(4).Range.Start, _
        End:=docActive.Paragraphs(8).Range.End)

With rngTemp.ListFormat
    .ApplyOutlineNumberDefault
    .ListIndent
End With
```



ListOutdent Method

-
Decreases the list level of the paragraphs in the range for the specified **ListFormat** object, in increments of one level.

expression.**ListOutdent**

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

Example

This example reduces the indent of each paragraph in first list in the active document by one level.

```
ActiveDocument.Lists(1).Range.ListFormat.ListOutdent
```

This example formats paragraphs four through eight in the active document as an outline-numbered list, indents the paragraphs one level, and then removes the indent from the first paragraph in the list.

```
Dim docActive As Document
Dim rngTemp As Range

Set docActive = ActiveDocument

Set rngTemp = _
    docActive.Range( _
        Start:=docActive.Paragraphs(4).Range.Start, _
        End:=docActive.Paragraphs(8).Range.End)

With rngTemp.ListFormat
    .ApplyOutlineNumberDefault
    .ListIndent
End With

docActive.Paragraphs(4).Range.ListFormat.ListOutdent
```



LookupNameProperties Method

↳ [Show All](#)

-

▶ [LookupNameProperties method as it applies to the **Application** object.](#)

Looks up a name in the global address book list and displays the **Properties** dialog box, which includes information about the specified name. If this method finds more than one match, it displays the **Check Names** dialog box.

expression.**LookupNameProperties**(*Name*)

expression Required. An expression that returns an [Application](#) object.

Name Required **String**. A name in the global address book.

▶ [LookupNameProperties method as it applies to the **Range** object.](#)

Looks up a name in the global address book list and displays the **Properties** dialog box, which includes information about the specified name. If this method finds more than one match, it displays the **Check Names** dialog box.

expression.**LookupNameProperties**

expression Required. An expression that returns a [Range](#) object.

Example

▶ [As it applies to the **Application** object.](#)

This example looks up the name Don Funk in the address book and displays the **Properties** dialog box for Don Funk.

```
Application.LookupNameProperties Name:="Don Funk"
```

▶ [As it applies to the **Range** object.](#)

This example looks up the selected name in the address book and displays the **Properties** dialog box for that person.

```
Selection.Range.LookupNameProperties
```



LtrPara Method

-
Sets the reading order and alignment of the specified paragraphs to left-to-right.

expression.**LtrPara**

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

Remarks

For all selected paragraphs, this method sets the reading order to left-to-right. If a paragraph with a right-to-left reading order is also right-aligned, this method reverses its reading order and sets its paragraph alignment to left-aligned.

Use the [ReadingOrder](#) property to change the reading order without affecting paragraph alignment.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the reading order and alignment of the current selection to left-to-right if the selection is styled as "Normal."

```
If Selection.Style = "Normal" Then _  
    Selection.LtrPara
```



↳ [Show All](#)

LtrRun Method

-
Sets the reading order and alignment of the specified [run](#) to left-to-right.

expression.LtrRun

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

Remarks

For the specified run, this method sets the reading order to left-to-right. If a paragraph in the run with a right-to-left reading order is also right-aligned, this method reverses its reading order and sets its paragraph alignment to left-aligned.

Use the [ReadingOrder](#) property to change the reading order without affecting paragraph alignment.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the reading order and alignment of the specified run to left-to-right if the selection is styled as "Normal."

```
If Selection.Style = "Normal" Then _  
    Selection.LtrRun
```



MakeCompatibilityDefault Method

-

Sets the compatibility options on the **Compatibility** tab in the **Options** dialog box (**Tools** menu) as the default settings for new documents.

expression.**MakeCompatibilityDefault**

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

Example

This example sets a few compatibility options for the active document and then makes the current compatibility options the default settings.

```
With ActiveDocument
    .Compatibility(wdSuppressSpBfAfterPgBrk) = True
    .Compatibility(wdExpandShiftReturn) = True
    .Compatibility(wdUsePrinterMetrics) = True
    .Compatibility(wdNoLeading) = False
    .MakeCompatibilityDefault
End With
```



ManualHyphenation Method

-

Initiates manual hyphenation of a document, one line at a time. The user is prompted to accept or decline suggested hyphenations.

expression.**ManualHyphenation**

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

Example

This example starts manual hyphenation of the active document.

```
ActiveDocument.ManualHyphenation
```

This example sets hyphenation options and then starts manual hyphenation of MyDoc.doc.

```
With Documents("MyDoc.doc")  
    .HyphenationZone = InchesToPoints(0.25)  
    .HyphenateCaps = False  
    .ManualHyphenation  
End With
```



MarkAllCitations Method

Inserts a TA (Table of Authorities Entry) field after all instances of the *ShortCitation* text.

expression.**MarkAllCitations**(*ShortCitation*, *LongCitation*, *LongCitationAutoText*, *Category*)

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

ShortCitation Required **String**. The short citation for the entry as it will appear in the **Mark Citation** dialog box (**Insert** menu, **Index and Tables** command).

LongCitation Optional **Variant**. The long citation string for the entry as it will appear in the table of authorities.

LongCitationAutoText Optional **Variant**. The AutoText entry name that contains the text of the long citation as it will appear in the table of authorities.

Category Optional **Variant**. The category number to be associated with the entry: 1 corresponds to the first category in the **Category** box in the **Mark Citation** dialog box, 2 corresponds to the second category, and so on.

Example

This example marks all instances of "Forrester v. Craddock" in the active document with a TA field that references the "Forrester v. Craddock, 51 Wn. 2d 315 (1957)" citation.

```
ActiveDocument.TablesOfAuthorities.MarkAllCitations _  
    ShortCitation:="Forrester v. Craddock", Category:=1, _  
    LongCitation:="Forrester v. Craddock, 51 Wn. 2d 315 (1957)"
```



MarkAllEntries Method

Inserts an XE (Index Entry) field after all instances of the text in **Range**.

expression.**MarkAllEntries**(**Range**, **Entry**, **EntryAutoText**, **CrossReference**, **CrossReferenceAutoText**, **BookmarkName**, **Bold**, **Italic**)

expression Required. An expression that returns an **Indexes** object.

Range Required **Range** object. The range whose text is marked with an XE field throughout the document.

Entry Optional **Variant**. The text you want to appear in the index, in the form *MainEntry[:Subentry]*.

EntryAutoText Optional **Variant**. The AutoText entry that contains the text you want to appear in the index (if this argument is specified, **Entry** is ignored).

CrossReference Optional **Variant**. A cross-reference that will appear in the index.

CrossReferenceAutoText Optional **Variant**. The name of the AutoText entry that contains the text for a cross-reference (if this argument is specified, **CrossReference** is ignored).

BookmarkName Optional **Variant**. The bookmark name that marks the range of pages you want to appear in the index. If this argument is omitted, the number of the page that contains the XE field appears in the index.

Bold Optional **Variant**. **True** to add bold formatting to page numbers for index entries.

Italic Optional **Variant**. **True** to add italic formatting to page numbers for index entries.

Example

This example marks the selected text with TA fields throughout the active document and then updates the first index in the document. The entry text in the index matches the selected text.

```
If Selection.Type = wdSelectionNormal Then
    ActiveDocument.Indexes.MarkAllEntries _
        Range:=Selection.Range, _
        Entry:=Selection.Range.Text, Italic:=True
    ActiveDocument.Indexes(1).Update
End If
```



MarkCitation Method

Inserts a TA (Table of Authorities Entry) field and returns the field as a **Field** object.

expression.**MarkCitation**(*Range*, *ShortCitation*, *LongCitation*, *LongCitationAutoText*, *Category*)

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

Range Required **Range** object. The location of the table of authorities entry. The TA field is inserted after **Range**.

ShortCitation Required **String**. The short citation for the entry as it will appear in the **Mark Citation** dialog box (**Insert** menu, **Index and Tables** command).

LongCitation Optional **Variant**. The long citation for the entry as it will appear in the table of authorities.

LongCitationAutoText Optional **Variant**. The name of the AutoText entry that contains the text of the long citation as it will appear in the table of authorities.

Category Optional **Variant**. The category number to be associated with the entry: 1 corresponds to the first category in the **Category** box in the **Mark Citation** dialog box, 2 corresponds to the second category, and so on.

Example

This example inserts a table of authorities entry (a TA field) that references the selected text. The long citation text is set to "Forrester v. Craddock" and the category is set to Other Cases.

```
ActiveDocument.TablesOfAuthorities.MarkCitation _  
    Range:=Selection.Range, ShortCitation:=Selection.Range.Text, _  
    LongCitation:="Forrester v. Craddock", Category:=1
```

This example inserts a table of authorities entry that references the selected text. The entry text that appears in the table of authorities is the text typed into the input box and the category is set to Other Authorities.

```
Dim strCitation As String
```

```
strCitation = InputBox("Type citation text")  
ActiveDocument.TablesOfAuthorities.MarkCitation _  
    Range:=Selection.Range, ShortCitation:=Selection.Range.Text, _  
    LongCitation:=strCitation, Category:=3
```



↳ [Show All](#)

MarkEntry Method

▶ [MarkEntry method as it applies to the **Indexes** object.](#)

Inserts an XE (Index Entry) field after the specified range. The XE field is returned as a **Field** object.

expression.**MarkEntry**(*Range*, *Entry*, *EntryAutoText*, *CrossReference*, *CrossReferenceAutoText*, *BookmarkName*, *Bold*, *Italic*, *Reading*)

expression Required. An expression that returns an **Indexes** object.

Range Required **Range** object. The location of the entry. The XE field is inserted after **Range**.

Entry Optional **Variant**. The text that appears in the index. To indicate a subentry, include the main entry text and the subentry text, separated by a colon (:) (for example, "Introduction:The Product").

EntryAutoText Optional **Variant**. The AutoText entry name that includes text for the index, table of figures, or table of contents (**Entry** is ignored).

CrossReference Optional **Variant**. A cross-reference that will appear in the index (for example, "See Apples").

CrossReferenceAutoText Optional **Variant**. The AutoText entry name that contains the text for a cross-reference (**CrossReference** is ignored).

BookmarkName Optional **Variant**. The name of the bookmark that marks the range of pages you want to appear in the index. If this argument is omitted, the number of the page containing the XE field appears in the index.

Bold Optional **Variant**. **True** to add bold formatting to the entry page numbers in the index.

Italic Optional **Variant**. **True** to add italic formatting to the entry page numbers

in the index.

Reading Optional **Variant**.

► [MarkEntry](#) method as it applies to the [TablesOfContents](#) and [TablesOfFigures](#) objects.

Inserts a TC (Table of Contents Entry) field after the specified range. The TC field is returned as a **Field** object.

expression.**MarkEntry**(**Range**, **Entry**, **EntryAutoText**, **TableID**, **Level**)

expression Required. An expression that returns a **TablesOfContents** or **TablesOfFigures** object.

Range Required **Range** object. The location of the entry. The TC field is inserted after **Range**.

Entry Optional **Variant**. The text that appears in the table of contents or table of figures. To indicate a subentry, include the main entry text and the subentry text, separated by a colon (:) (for example, "Introduction:The Product").

EntryAutoText Optional **Variant**. The AutoText entry name that includes text for the index, table of figures, or table of contents (**Entry** is ignored).

TableID Optional **Variant**. A one-letter identifier for the table of figures or table of contents item (for example, "i" for an "illustration").

Level Optional **Variant**. A level for the entry in the table of contents or table of figures.

Example

▶ [As it applies to the **Indexes** object.](#)

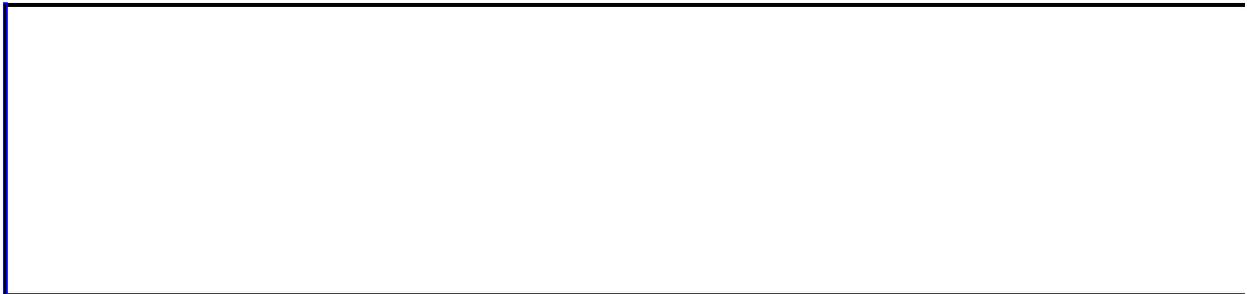
This example inserts an index entry after the selection in the active document. The subentry text is the text from the selection.

```
If Selection.Type = wdSelectionNormal Then
    ActiveDocument.Indexes.MarkEntry Range:=Selection.Range, _
        Entry:="Introduction:" & Selection.Range.Text, Italic:=TrueE
```

▶ [As it applies to the **Table of Contents** object.](#)

This example inserts a table of contents entry that references the selected text. The text typed in the input box appears in the table of contents. A table of contents that uses fields is then added at the beginning of the active document.

```
entryText = InputBox("Type entry text")
ActiveDocument.TablesOfContents.MarkEntry _
    Range:=Selection.Range, Entry:=entryText
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.TablesOfContents.Add _
    Range:=myRange, UseFields:=True, _
    UseHeadingStyles:=False
```



↳ [Show All](#)

Merge Method

▶ [Merge method as it applies to the **Subdocuments** object.](#)

Merges the specified subdocuments of a master document into a single subdocument.

expression.Merge(FirstSubdocument, LastSubdocument)

expression Required. An expression that returns one of the above objects.

FirstSubdocument Optional **Variant**. The path and file name of the original document you want to merge revisions with.

LastSubdocument Optional **Variant**. The last subdocument in a range of subdocuments to be merged.

▶ [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 one of the above objects.

MergeTo Required **Cell** object. The cell to be merged with.

▶ [Merge method as it applies to the **Document** object.](#)

Merges the changes marked with revision marks from one document to another.

expression.Merge(Name, MergeTarget, DetectFormatChanges,

UseFormattingFrom, AddToRecentFiles)

expression Required. An expression that returns one of the above objects.

Name Required **String**.

MergeTarget Optional [WdMergeTarget](#).

WdMergeTarget can be one of these WdMergeTarget constants.

wdMergeTargetCurrent *default*

wdMergeTargetSelected

wdMergeTargetNew

DetectFormatChanges Optional **Boolean**.

UseFormattingFrom Optional [WdUseFormattingFrom](#).

WdUseFormattingFrom can be one of these WdUseFormattingFrom constants.

wdFormattingFromPrompt *default*

wdFormattingFromCurrent

wdFormattingFromSelected

AddToRecentFiles Optional **Boolean**.

▶ [Merge method as it applies to the Cells object.](#)

Merges the specified table cells with one another. The result is a single table cell.

expression.**Merge**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Cell** object.](#)

This example merges the first two cells in table one in the active document with one another and then removes the table borders.

```
If ActiveDocument.Tables.Count >= 1 Then
    With ActiveDocument.Tables(1)
        .Cell(Row:=1, Column:=1).Merge _
            MergeTo:=.Cell(Row:=1, Column:=2)
        .Borders.Enable = False
    End With
End If
```

▶ [As it applies to the **Document** object.](#)

This example merges changes from Sales1.doc into Sales2.doc (the active document).

```
If InStr(1, ActiveDocument.Name, "sales2.doc", 1) Then _
    ActiveDocument.Merge FileName:="C:\Docs\Sales1.doc"
```

▶ [As it applies to the **Cells** object.](#)

This example merges the cells in row one of the selection into a single cell and then applies shading to the row.

```
If Selection.Information(wdWithInTable) = True Then
    Set myrow = Selection.Rows(1)
    myrow.Cells.Merge
    myrow.Shading.Texture = wdTexture10Percent
End If
```

▶ [As it applies to the **Subdocuments** object.](#)

This example merges the first and second subdocuments in the active document into one subdocument.

```
If ActiveDocument.Subdocuments.Count >= 2 Then
    Set aDoc = ActiveDocument
    aDoc.Subdocuments.Merge _
```

```
FirstSubdocument:=aDoc.Subdocuments(1), _  
LastSubdocument:=aDoc.Subdocuments(2)
```

```
End If
```



MillimetersToPoints Method

-
Converts a measurement from millimeters to points (1 mm = 2.85 points).
Returns the converted measurement as a **Single**.

expression.**MillimetersToPoints**(*Millimeters*)

expression Optional. An expression that returns an **Application** object.

Millimeters Required **Single**. The millimeter value to be converted to points.

Example

This example sets the hyphenation zone in the active document to 8.8 millimeters.

```
ActiveDocument.HyphenationZone = MillimetersToPoints(8.8)
```

This example expands the spacing of the selected characters to 2.8 points.

```
Selection.Font.Spacing = MillimetersToPoints(1)
```



↳ [Show All](#)

ModifyEnclosure Method

Adds, modifies, or removes an enclosure around the specified character or characters.

expression.**ModifyEnclosure**(*Style*, *Symbol*, *EnclosedText*)

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

Style Required **Variant**. The style of the enclosure. Can be any [WdEncloseStyle](#) constant.

WdEncloseStyle can be one of these WdEncloseStyle constants.

wdEncloseStyleLarge

wdEncloseStyleNone

wdEncloseStyleSmall

Symbol Optional **Variant**. The symbol in which to enclose the specified range. Can be any [WdEnclosureType](#) constant.

WdEnclosureType can be one of these WdEnclosureType constants.

wdEnclosureCircle Default.

wdEnclosureDiamond

wdEnclosureSquare

wdEnclosureTriangle

EnclosedText Optional **Variant**. The characters that you want to enclose. If you include this argument, Microsoft Word replaces the specified range with the enclosed characters. If you don't specify text to enclose, Microsoft Word encloses all text in the specified range.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example replaces the current selection with the number 25 enclosed in a circle.

```
Selection.Range.ModifyEnclosure wdEncloseStyleLarge, _  
    wdEnclosureCircle, "25"
```



↳ [Show All](#)

Move Method

▶ [Move method as it applies to the **Range** and **Selection** objects.](#)

Collapses the specified range or selection to its start or end position and then moves the collapsed object by the specified number of units. This method returns a **Long** value that indicates the number of units by which the object was actually moved, or it returns 0 (zero) if the move was unsuccessful.

expression.**Move**(*Unit*, *Count*)

expression Required. An expression that returns one of the above objects.

Unit Optional **Variant**. The unit by which the collapsed range or selection is to be moved. Can be one of the following **WdUnits** constants: **wdCharacter**, **wdWord**, **wdSentence**, **wdParagraph**, **wdSection**, **wdStory**, **wdCell**, **wdColumn**, **wdRow**, or **wdTable**. If *expression* returns a [Selection](#) object, you can also use **wdLine**. The default value is **wdCharacter**.

Count Optional **Variant**. The number of units by which the specified range or selection is to be moved. If **Count** 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 **Count** is a negative number, the object is collapsed to its start position and moved backward by the specified number of units. The default value is 1. You can also control the collapse direction by using the **Collapse** method before using the **Move** method. If the range or selection is in the middle of a unit or isn't collapsed, moving it to the beginning or end of the unit counts as moving it one full unit.

Remarks

The start and end positions of a collapsed range or selection are equal.

Applying the **Move** method to a range doesn't rearrange text in the document. Instead, it redefines the range to refer to a new location in the document.

If you apply the **Move** method to any range other than a [Range](#) object variable (for example, `Selection.Paragraphs(3).Range.Move`), the method has no effect.

Moving a **Selection** object collapses the selection and moves the insertion point either forward or backward in the document.

► [Move method as it applies to the **Application** and **Task** objects.](#)

Positions a task window or 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.

► [Move method as it applies to the **StyleSheet** object.](#)

Moves a style sheet's order of precedence.

expression.**Move**(**Precedence**)

expression Required. An expression that returns a [StyleSheet](#) object.

Precedence Required [WdStyleSheetPrecedence](#). The precedence level.

WdStyleSheetPrecedence can be one of these WdStyleSheetPrecedence constants.

wdStyleSheetPrecedenceHigher

wdStyleSheetPrecedenceHighest
wdStyleSheetPrecedenceLower
wdStyleSheetPrecedenceLowest

Example

▶ [As it applies to the **Application** object.](#)

This example starts the Calculator application (Calc.exe) and uses the **Move** method to reposition the application window.

```
Shell "Calc.exe"  
With Tasks("Calculator")  
    .WindowState = wdWindowStateNormal  
    .Move Top:=50, Left:=50  
End With
```

▶ [As it applies to the **Range** object.](#)

This example sets Range1 to the first paragraph in the active document and then moves the range forward three paragraphs. After this macro is run, the insertion point is positioned at the beginning of the fourth paragraph.

```
Set Range1 = ActiveDocument.Paragraphs(1).Range  
With Range1  
    .Collapse Direction:=wdCollapseStart  
    .Move Unit:=wdParagraph, Count:=3  
    .Select  
End With
```

▶ [As it applies to the **Selection** object.](#)

This example moves the selection two words to the right and positions the insertion point after the second word's trailing space. If the move is unsuccessful, a message box indicates that the selection is at the end of the document.

```
If Selection.StoryType = wdMainTextStory Then  
    wUnits = Selection.Move(Unit:=wdWord, Count:=2)  
    If wUnits < 2 Then _  
        MsgBox "Selection is at the end of the document"  
End If
```

This example moves the selection forward three cells in the table.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.Move Unit:=wdCell, Count:=3
```

End If



MoveDown Method

Moves the selection down and returns the number of units it's been moved.

Note The **wdWindow** constant can be used to move to the top or bottom of the active window. Regardless of the value of **Count** (greater than 1 or less than -1), the **wdWindow** constant moves only one unit. Use the **wdScreen** constant to move more than one screen.

expression.MoveDown(**Unit**, **Count**, **Extend**)

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

Unit Optional [WdUnits](#). The unit by which the selection is to be moved.

Can be one of the following **WdUnits** constants.

wdLine

wdParagraph

wdWindow

wdScreen

The default value is **wdLine**.

Count Optional **Variant**. The number of units the selection is to be moved. The default value is 1.

Extend Optional **Variant**. Can be either **wdMove** or **wdExtend**. If **wdMove** is used, the selection is collapsed to the end point and moved down. If **wdExtend** is used, the selection is extended down. The default value is **wdMove**.

Example

This example extends the selection down one line.

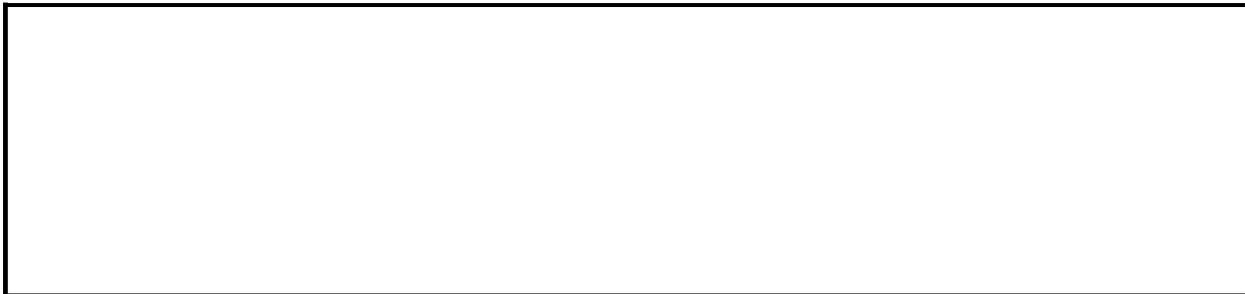
```
Selection.MoveDown Unit:=wdLine, Count:=1, Extend:=wdExtend
```

This example moves the selection down three paragraphs. If the move is successful, "Company" is inserted at the insertion point.

```
unitsMoved = Selection.MoveDown(Unit:=wdParagraph, _  
    Count:=3, Extend:=wdMove)  
If unitsMoved = 3 Then Selection.Text = "Company"
```

This example displays the current line number, moves the selection down three lines, and displays the current line number again.

```
MsgBox "Line " & Selection.Information(wdFirstCharacterLineNumber)  
Selection.MoveDown Unit:=wdLine, Count:=3, Extend:=wdMove  
MsgBox "Line " & Selection.Information(wdFirstCharacterLineNumber)
```



MoveEnd Method

Moves the ending character position of a range or selection. This method returns an integer that indicates the number of units the range or selection actually moved, or it returns 0 (zero) if the move was unsuccessful.

expression.**MoveEnd**(Unit, Count)

expression Required. An expression that returns a **Range** or **Selection** object.

Unit Optional [WdUnits](#). The unit by which to move the ending character position.

Can be one of the following **WdUnits** constants

wdCharacter

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable.

If *expression* returns a **Selection** object, **wdLine** can also be used. The default

value is **wdCharacter**.

Count Optional **Variant**. 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. The default value is 1.

Example

This example moves the end of the selection one character backward (the selection size is reduced by one character). A space is considered a character.

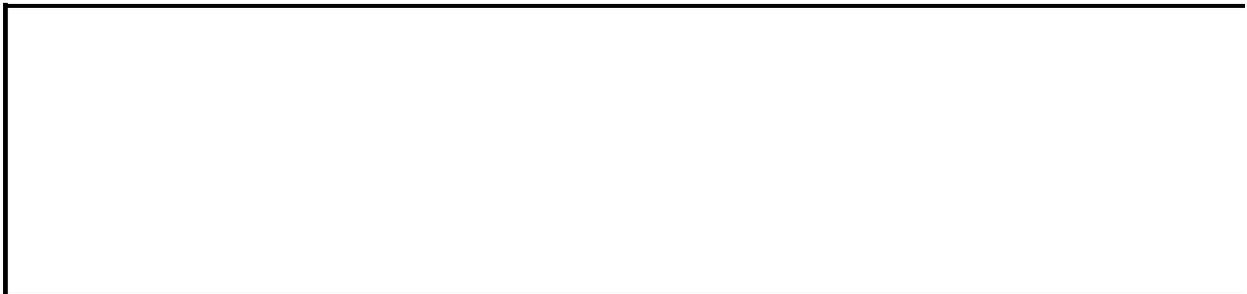
```
Selection.MoveEnd Unit:=wdCharacter, Count:=-1
```

This example moves the end of the selection to the end of the line (the selection is extended to the end of the line).

```
Selection.MoveEnd Unit:=wdLine, Count:=1
```

This example sets myRange to be equal to the second word in the active document. The **MoveEnd** method is used to move the ending position of myRange (a range object) forward one word. After this macro is run, the second and third words in the document are selected.

```
If ActiveDocument.Words.Count >= 3 Then  
    Set myRange = ActiveDocument.Words(2)  
    With myRange  
        .MoveEnd Unit:=wdWord, Count:=1  
        .Select  
    End With  
End If
```



MoveEndUntil Method

-

Moves the end position of the specified range or selection until any of the specified characters are found in the document. If the movement is forward in the document, the range or selection is expanded.

Remarks

This method returns the number of characters by which the end position of the specified range or selection was moved, as a **Long** value. If **Count** is greater than 0 (zero), this method returns the number of characters moved plus 1. If **Count** is less than 0 (zero), this method returns the number of characters moved minus 1. If no **Cset** characters are found, the range or selection isn't changed and the method returns 0 (zero). If the end position is moved backward to a point that precedes the original start position, the start position is set to the new ending position.

expression.**MoveEndUntil(Cset, Count)**

expression Required. An expression that returns a **Range** or **Selection** object.

Cset Required **Variant**. One or more characters. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the specified range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the range or selection is moved forward in the document. If it's a negative number, the range or selection is moved backward. The default value is **wdForward**.

Example

This example extends the selection forward in the document until the letter "a" is found. The example then expands the selection by one character to include the letter "a".

With Selection

```
.MoveEndUntil Cset:="a", Count:=wdForward
```

```
.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
```

End With

This example extends the selection forward in the document until a tab is found. If a tab character isn't found in the next 100 characters, the selection isn't moved.

```
char = Selection.MoveEndUntil(Cset:=vbTab, Count:=100)
```

```
If char = 0 Then StatusBar = "Selection not moved"
```



MoveEndWhile Method

-

Moves the ending character position of a range or selection while any of the specified characters are found in the document.

Remarks

While any character in **Cset** is found, the end position of the specified range or selection is moved. This method returns the number of characters that the end position of the range or selection moved as a **Long** value. If no **Cset** characters are found, the range or selection isn't changed and the method returns 0 (zero). If the end position is moved backward to a point that precedes the original start position, the start position is set to the new end position.

expression.**MoveEndWhile**(**Cset**, **Count**)

expression Required. An expression that returns a **Range** or **Selection** object.

Cset Required **Variant**. One or more characters. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the range or selection is moved forward in the document. If it's a negative number, the range or selection is moved backward. The default value is **wdForward**.

Example

This example moves the end position of the selection forward while the space character is found.

```
Selection.MoveEndWhile Cset:=" ", Count:=wdForward
```

This example moves the end position of the selection forward while *Count* is less than or equal to 10 and any letter from "a" through "h" is found.

```
Selection.MoveEndWhile Cset:="abcdefgh", Count:=10
```



MoveLeft Method

Moves the selection to the left and returns the number of units it's been moved.

expression.**MoveLeft**(*Unit*, *Count*, *Extend*)

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

Unit Optional [WdUnits](#). The unit by which the selection is to be moved.

Can be one of the following **WdUnits** constants.

wdCell

wdCharacter

wdWord

wdSentence

The default value is **wdCharacter**.

Count Optional **Variant**. The number of units the selection is to be moved. The default value is 1.

Extend Optional **Variant**. Can be either **wdMove** or **wdExtend**. If **wdMove** is used, the selection is collapsed to the end point and moved to the left. If **wdExtend** is used, the selection is extended to the left. The default value is **wdMove**.

Remarks

When the *Unit* is **wdCell**, the *Extend* argument will only be **wdMove**.

Example

This example moves the selection one character to the left. If the move is successful, **MoveLeft** returns 1.

```
If Selection.MoveLeft = 1 Then MsgBox "Move was successful"
```

This example enables field shading for the selected field, inserts a DATE field, and then moves the selection left to select the field.

```
ActiveDocument.ActiveWindow.View.FieldShading = _  
    wdFieldShadingWhenSelected  
With Selection  
    .Fields.Add Range:=Selection.Range, Type:=wdFieldDate  
    .MoveLeft Unit:=wdWord, Count:=1  
End With
```

This example moves the selection to the previous table cell.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.MoveLeft Unit:=wdCell, Count:=1, Extend:=wdMove  
End If
```



↳ [Show All](#)

MoveNode Method

Moves a diagram node and any of its child nodes within a diagram.

expression.**MoveNode**(*TargetNode*, *Pos*)

expression Required. An expression that returns a [DiagramNode](#) object.

TargetNode Required **DiagramNode** object. The diagram node where the specified node will be moved.

Pos Required [MsoRelativeNodePosition](#). Specifies where the node will be added relative to **TargetNode**.

MsoRelativeNodePosition can be one of these MsoRelativeNodePosition constants.

msoAfterLastSibling

msoAfterNode

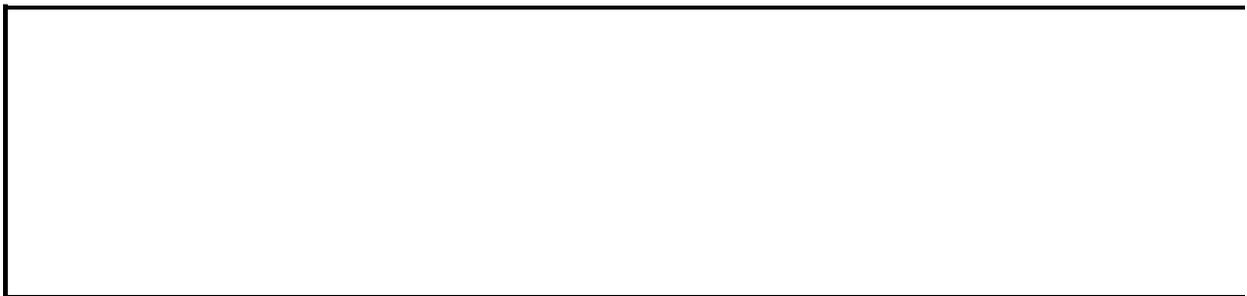
msoBeforeFirstSibling

msoBeforeNode

Example

The following example moves the second diagram node of a newly-created diagram to the last node.

```
Sub MoveDiagramNode()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add four child nodes to the pyramid diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    'Move the second node after the fourth node  
    dgnNode.Diagram.Nodes(2).MoveNode _  
        TargetNode:=dgnNode.Diagram.Nodes(4), _  
        Pos:=msoAfterLastSibling  
End Sub
```



MoveRight Method

Moves the selection to the right and returns the number of units it's been moved.

expression.**MoveRight**(*Unit*, *Count*, *Extend*)

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

Unit Optional [WdUnits](#). The unit by which the selection is to be moved.

Can be one of the following **WdUnits** constants.

wdCell

wdCharacter

wdWord

wdSentence

The default value is **wdCharacter**.

Count Optional **Variant**. The number of units the selection is to be moved. The default value is 1.

Extend Optional **Variant**. Can be either **wdMove** or **wdExtend**. If **wdMove** is used, the selection is collapsed to the end point and moved to the right. If **wdExtend** is used, the selection is extended to the right. The default value is **wdMove**.

Remarks

When the *Unit* is **wdCell**, the *Extend* argument will only be **wdMove**.

Example

This example moves the selection before the previous field and then selects the field.

```
With Selection
    Set MyRange = .GoTo(wdGoToField, wdGoToPrevious)
    .MoveRight Unit:=wdWord, Count:=1, Extend:=wdExtend
    If Selection.Fields.Count = 1 Then Selection.Fields(1).Update
End With
```

This example moves the selection one character to the right. If the move is successful, **MoveRight** returns 1.

```
If Selection.MoveRight = 1 Then MsgBox "Move was successful"
```

This example moves the selection to the next table cell.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.MoveRight Unit:=wdCell, Count:=1, Extend:=wdMove
End If
```



MoveStart Method

Moves the start position of the specified range or selection. This method returns an integer 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*, *Count*)

expression Required. An expression that returns a **Range** or **Selection** object.

Unit Optional [WdUnits](#). The unit by which start position of the specified range or selection is to be moved.

Can be one of the following **WdUnits** constants.

wdCharacter

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable

If *expression* returns a **Selection** object, you can also use **wdLine**. The default value is **wdCharacter**.

Count Optional **Variant**. The maximum number of units by which the specified range or selection is to be moved. If **Count** is a positive number, the start position of the range or selection is moved forward in the document. If it's a negative number, the start position is moved backward. If the start position is moved forward to a position beyond the end position, the range or selection is collapsed and both the start and end positions are moved together. The default value is 1.

Example

This example moves the start position of the selection one character forward (the selection size is reduced by one character). Note that a space is considered a character.

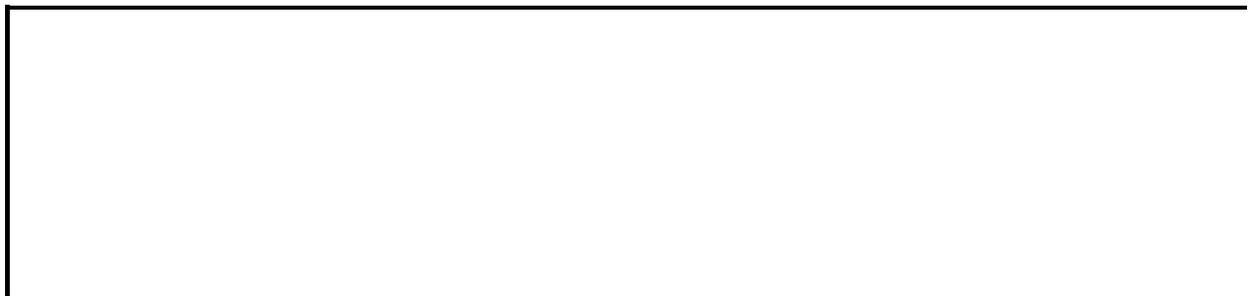
```
Selection.MoveStart Unit:=wdCharacter, Count:=1
```

This example moves the start position of the selection to the beginning of the line (the selection is extended to the start of the line).

```
Selection.MoveStart Unit:=wdLine, Count:=-1
```

This example sets myRange to be equal to the second word in the active document. The example uses the **MoveStart** method to move the start position of myRange (a **Range** object) backward one word. After this macro is run, the first and second words in the document are selected.

```
If ActiveDocument.Words.Count >= 2 Then  
    Set myRange = ActiveDocument.Words(2)  
    With myRange  
        .MoveStart Unit:=wdWord, Count:=-1  
        .Select  
    End With  
End If
```



MoveStartUntil Method

-

Moves the start position of the specified range or selection until one of the specified characters is found in the document. If the movement is backward through the document, the range or selection is expanded.

Remarks

This method returns the number of characters by which the start position of the specified range or selection moved, as a **Long** value. If **Count** is greater than 0 (zero), this method returns the number of characters moved plus 1. If **Count** is less than 0 (zero), this method returns the number of characters moved minus 1. If no **Cset** characters are found, the specified range or selection isn't changed and the method returns 0 (zero). If the start position is moved forward to a point beyond the end position, the range or selection is collapsed and both the start and end positions are moved together.

expression.**MoveStartUntil(Cset, Count)**

expression Required. An expression that returns an object in the **Applies To** list.

Cset Required **Variant**. One or more characters. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the specified range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the range or selection is moved forward in the document. If it's a negative number, the range or selection is moved backward. The default value is **wdForward**.

Example

This example extends the selection backward until a capital "I" is found.

```
Selection.MoveStartUntil Cset:="I", Count:=wdBackward
```

If there's a dollar sign character (\$) in the first paragraph in the selection, this example moves myRange just before the dollar sign.

```
Set myRange = Selection.Paragraphs(1).Range  
leng = myRange.End - myRange.Start  
myRange.Collapse Direction:=wdCollapseStart  
myRange.MoveStartUntil Cset:="$", Count:=leng
```



MoveStartWhile Method

-

Moves the start position of the specified range or selection while any of the specified characters are found in the document.

Remarks

While any character in **Cset** is found, the start position of the range or selection is moved. This method returns the number of characters that the start position of the range or selection moved as a **Long** value. If not **Cset** characters are found, the range or selection isn't changed and the method returns 0 (zero). If the start position is moved forward to a position beyond the original end position, the end position is set to the new start position.

expression.**MoveStartWhile(Cset, Count)**

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

Cset Required **Variant**. One or more characters. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the specified range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the range or selection is moved forward in the document. If it's a negative number, the range or selection is moved backward. The default value is **wdForward**.

Example

This example moves the start position of the selection backward through the document while the space character is found.

```
Selection.MoveStartWhile Cset:=" ", Count:=wdBackward
```

This example moves the start position of the selection backward through the document while **Count** is less than or equal to 10 and any letter from "a" through "h" is found.

```
Selection.MoveStartWhile Cset:="abcdefgh", Count:=-10
```



MoveUntil Method

Moves the specified range or selection until one of the specified characters is found in the document.

expression.**MoveUntil**(*Cset*, *Count*)

expression Required. An expression that returns a **Range** of **Selection** object.

Cset Required **Variant**. One or more characters. If any character in **Cset** is found before the **Count** value expires, the specified range or selection is positioned as an insertion point immediately before that character. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the specified range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the range or selection is moved forward in the document, beginning at the end position. If it's a negative number, the range or selection is moved backward, beginning at the start position. The default value is **wdForward**.

Remarks

This method returns the number of characters by which the specified range or selection was moved, as a **Long** value. If **Count** is greater than 0 (zero), this method returns the number of characters moved plus one. If **Count** is less than 0 (zero), this method returns the number of characters moved minus one. If no **Cset** characters are found, the range or selection isn't not changed and the method returns 0 (zero).

Example

This example moves myRange forward through the next 100 characters in the document until the character "t" is found.

```
Set myRange = ActiveDocument.Words(1)
myRange.MoveUntil Cset:="t", Count:=100
```

This example moves the selection forward to the end of the active paragraph and then displays the number of characters by which the selection was moved.

```
x = Selection.MoveUntil(Cset:=Chr$(13), Count:=wdForward)
MsgBox x-1 & " character positions were moved"
```



MoveUp Method

Moves the selection up and returns the number of units it's been moved.

Note The **wdWindow** constant can be used to move to the top or bottom of the active window. Regardless of the value of **Count** (greater than 1 or less than – 1), the **wdWindow** constant moves only one unit. Use the **wdScreen** constant to move more than one screen.

expression.MoveUp(Unit, Count, Extend)

expression Required. An expression that returns an object in the Applies To list.

Unit Optional **VARIANT**. The unit by which to move the selection. Can be one of the following **WdUnits** constants: **wdLine**, **wdParagraph**, **wdWindow** or **wdScreen**. The default value is **wdLine**.

Count Optional **VARIANT**. The number of units the selection is to be moved. The default value is 1.

Extend Optional **VARIANT**. Can be either **wdMove** or **wdExtend**. If **wdMove** is used, the selection is collapsed to the end point and moved up. If **wdExtend** is used, the selection is extended up. The default value is **wdMove**.

Example

This example moves the selection to the beginning of the previous paragraph.

```
Selection.MoveRight  
Selection.MoveUp Unit:=wdParagraph, Count:=2, Extend:=wdMove
```

This example displays the current line number, moves the selection up three lines, and displays the current line number again.

```
MsgBox "Line " & Selection.Information(wdFirstCharacterLineNumber)  
Selection.MoveUp Unit:=wdLine, Count:=3, Extend:=wdMove  
MsgBox "Line " & Selection.Information(wdFirstCharacterLineNumber)
```



MoveWhile Method

-

Moves the specified range or selection while any of the specified characters are found in the document.

Remarks

While any character in **Cset** is found, the specified range or selection is moved. The resulting **Range** or **Selection** object is positioned as an insertion point after whatever **Cset** characters were found. This method returns the number of characters by which the specified range or selection was moved, as a **Long** value. If no **Cset** characters are found, the range or selection isn't changed and the method returns 0 (zero).

expression.**MoveWhile(Cset, Count)**

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

Cset Required **Variant**. One or more characters. This argument is case sensitive.

Count Optional **Variant**. The maximum number of characters by which the specified range or selection is to be moved. Can be a number or either the **wdForward** or **wdBackward** constant. If **Count** is a positive number, the specified range or selection is moved forward in the document, beginning at the end position. If it's a negative number, the range or selection is moved backward, beginning at the start position. The default value is **wdForward**.

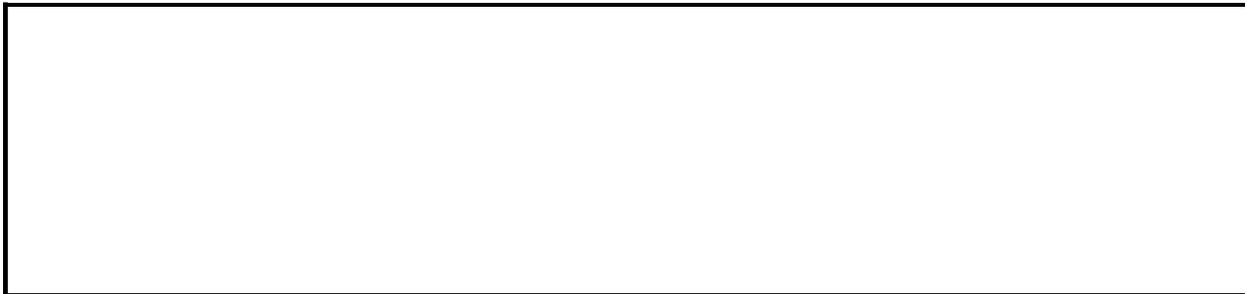
Example

This example moves the selection after consecutive tabs.

```
Selection.MoveWhile Cset:=vbTab, Count:=wdForward
```

This example moves aRange while any of the following (uppercase or lowercase) letters are found: "a", "t", or "i".

```
Set aRange = ActiveDocument.Characters(1)  
aRange.MoveWhile Cset:="atiATI", Count:=wdForward
```



MSInfo Method

-
Starts the Microsoft System Information application if it's not running, or switches to it if it's already running.

expression.**MSInfo**

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

Example

This example starts or switches to the Microsoft System Information application.

System.**MSInfo**



New Method

-

Inserts an empty, 1-inch-square Word picture object surrounded by a border. This method returns the new graphic as an [InlineShape](#) object.

expression.**New**(*Range*)

expression Required. An expression that returns an **InlineShapes** object.

Range Required **Range** object. The location of the new graphic.

Example

This example inserts a new, empty picture in the active document and applies a shadow border around the picture.

```
Dim ishapeNew As InlineShape
```

```
Set ishapeNew = _  
    ActiveDocument.InlineShapes.New(Range:=Selection.Range)
```

```
ishapeNew.Borders.Shadow = True
```

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
```



NewFrameset Method

-
Creates a new frames page based on the specified pane.

expression.**NewFrameset**

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

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example opens a document named "Temp.doc" and then creates a new frames page whose only frame contains "Temp.doc".

```
Documents.Open "C:\Documents\Temp.doc"  
ActiveDocument.ActiveWindow.ActivePane.NewFrameset
```



NewWindow Method

-
Opens a new window with the same document as the specified window. Returns a **Window** object.

Note A colon (:) and a number appear in the window caption when more than one window is open for a document.

expression.**NewWindow**

expression Required. An expression that returns an **Application** or **Window** object.

Remarks

If the **NewWindow** method is used with the **Application** object, a new window is opened for the active window. The following two instructions are functionally equivalent.

```
Set myWindow = ActiveDocument.ActiveWindow.NewWindow  
Set myWindow = NewWindow
```

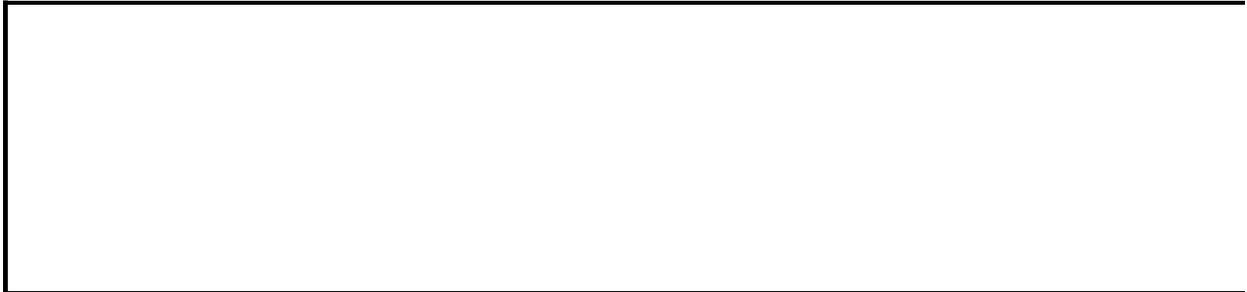
Example

This example posts a message that indicates the number of windows that exist before and after you open a new window for Document1.

```
MsgBox Windows.Count & " windows open"  
Windows("Document1").NewWindow  
MsgBox Windows.Count & " windows open"
```

This example opens a new window, arranges all the open windows, closes the new window, and then rearranges the open windows.

```
Set myWindow = NewWindow  
Windows.Arrange ArrangeStyle:=wdTiled  
myWindow.Close  
Windows.Arrange ArrangeStyle:=wdTiled
```



↳ [Show All](#)

Next Method

▶ [Next method as it applies to the **Paragraph** object.](#)

Returns a **Paragraph** object that represents the next paragraph.

expression.Next(**Count**)

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

Count Optional **Variant**. The number of paragraphs by which you want to move ahead. The default value is one.

▶ [Next method as it applies to the **Range** and **Selection** objects.](#)

Returns a **Range** object that represents the specified unit relative to the specified selection or range.

expression.Next(**Unit**, **Count**)

expression Required. An expression that returns one of the above objects.

Unit Optional **Variant**. The type of units by which to count. Can be any [WdUnits](#) constant.

WdUnits can be one of these WdUnits constants.

wdCharacter Default.

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable

wdLine Can be used if *expression* returns a **Selection** object.

Count Optional **Variant**. The number of units by which you want to move ahead. The default value is one.

Remarks

If the **Range** or **Selection** is just before the specified **Unit**, the **Range** or **Selection** is moved to the following unit. For example, if the **Selection** is just before a word, the following instruction moves the **Selection** forward to the following word.

```
Selection.Next(Unit:=wdWord, Count:=1).Select
```

▶ [Next method as it applies to the **Browser** object.](#)

Moves the selection to the next item indicated by the browser target. Use the [Target](#) property to change the browser target.

expression.**Next**

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

Example

▶ [As it applies to the **Browser** object.](#)

This example moves the insertion point just before the next comment reference marker in the active document.

```
With Application.Browser
    .Target = wdBrowseComment
    .Next
End With
```

▶ [As it applies to the **Paragraph** object](#)

This example inserts a number and a tab before the first nine paragraphs in the active document.

```
For n = 0 To 8
    Set myRange = ActiveDocument.Paragraphs(1).Next(Count:=n).Range
    myRange.Collapse Direction:=wdCollapseStart
    myRange.InsertAfter n + 1 & vbTab
Next n
```

This example selects the paragraph following the current selection.

```
Selection.Next(Unit:=wdParagraph, Count:=1).Select
```



NextCitation Method

-
Finds and selects the next instance of the text specified by *ShortCitation*.

expression.**NextCitation**(*ShortCitation*)

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

ShortCitation Required **String**. The text of the short citation.

Example

This example selects the next citation in the active document that begins with "in re".

```
ActiveDocument.TablesOfAuthorities.NextCitation _  
    ShortCitation:="in re"
```



NextField Method

-
Selects the next field. If a field is found, this method returns a **Field** object; if not, it returns **Nothing**.

expression.NextField

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

Example

This example updates the next field in the selection.

```
If Not (Selection.NextField Is Nothing) Then  
    Selection.Fields.Update  
End If
```

This example selects the next field in the selection, and if a field is found, displays a message in the status bar.

```
Set myField = Selection.NextField  
If Not (myField Is Nothing) Then StatusBar = "Field found"
```



NextHeaderFooter Method

-

If the selection is in a header, this method moves to the next header within the current section (for example, from an odd header to an even header) or to the first header in the following section. If the selection is in a footer, this method moves to the next footer.

Note If the selection is in the last header or footer in the last section of the document, or if it's not in a header or footer at all, an error occurs.

expression.**NextHeaderFooter**

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

Example

This example displays the first page header in the active document and then switches to the next header. The document needs to be at least two pages long.

```
ActiveDocument.PageSetup.DifferentFirstPageHeaderFooter = True
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .SeekView = wdSeekFirstPageHeader
    .NextHeaderFooter
End With
```



NextNode Method

Returns the next [DiagramNode](#) object in a collection of diagram nodes.

expression.**NextNode**

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

Remarks

Use the [PrevNode](#) method to return the previous **DiagramNode** object in a collection of diagram nodes.

Example

This example creates an organization chart and adds child nodes to the middle diagram node.

```
Sub AddChildrenToMiddle()  
    Dim dgnRoot As DiagramNode  
    Dim shpDiagram As Shape  
    Dim dgnNext As DiagramNode  
    Dim intCount As Integer  
  
    'Add organization chart to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add four child nodes to organization chart  
    Set dgnRoot = shpDiagram.DiagramNode.Children.AddNode  
  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next  
  
    'Access the node immediately following the  
    'first diagram node and add three child nodes  
    Set dgnNext = dgnRoot.Children.Item(1).NextNode  
  
    For intCount = 1 To 3  
        dgnNext.Children.AddNode  
    Next intCount  
  
End Sub
```



NextRevision Method

Locates and returns the next tracked change as a [Revision](#) object. The changed text becomes the current selection. Use the properties of the resulting **Revision** object to see what type of change it is, who made it, and so forth. Use the methods of the **Revision** object to accept or reject the change.

expression.**NextRevision**(*Wrap*)

expression Required. An expression that returns a [Selection](#) object.

Wrap Optional **Variant**. **True** to continue searching for a revision at the beginning of the document when the end of the document is reached. The default value is **False**.

Remarks

If there are no tracked changes to be found, the current selection remains unchanged.

Example

This example rejects the next tracked change found after the fifth paragraph in the active document. The `revTemp` variable is set to **Nothing** if a change is not found.

```
Dim rngTemp as Range
Dim revTemp as Revision

If ActiveDocument.Paragraphs.Count >= 5 Then
    Set rngTemp = ActiveDocument.Paragraphs(5).Range
    rngTemp.Select
    Set revTemp = Selection.NextRevision(Wrap:=False)
    If Not (revTemp Is Nothing) Then revTemp.Reject
End If
```

This example accepts the next tracked change found if the change type is inserted text.

```
Dim revTemp as Revision

Set revTemp = Selection.NextRevision(Wrap:=True)
If Not (revTemp Is Nothing) Then
    If revTemp.Type = wdRevisionInsert Then revTemp.Accept
End If
```

This example finds the next revision after the current selection made by the author of the document.

```
Dim revTemp as Revision
Dim strAuthor as String

strAuthor = ActiveDocument.BuiltInDocumentProperties(wdPropertyAuthor)

Do While True
    Set revTemp = Selection.NextRevision(Wrap:=False)
    If Not (revTemp Is Nothing) Then
        If revTemp.Author = strAuthor Then
            MsgBox Prompt:="Another revision by " & strAuthor & "!"
            Exit Do
        End If
    Else
        MsgBox Prompt:="No more revisions!"
    End If
End Do
```

```
Exit Do
End If
Loop
```



NextSubdocument Method

-

Moves the range or selection to the next subdocument. If there isn't another subdocument, an error occurs.

expression.**NextSubdocument**

expression Required. An expression that returns a **Range** or **Selection** object.

Example

This example switches the active document to master document view and selects the first subdocument.

```
If ActiveDocument.Subdocuments.Count >= 1 Then
    ActiveDocument.ActiveWindow.View.Type = wdMasterView
    Selection.HomeKey unit:=wdStory, Extend:=wdMove
    Selection.NextSubdocument
End If
```



This keyword is not implemented. It is reserved for future use.

OneColorGradient Method

Sets the specified fill to a one-color gradient.

expression.**OneColorGradient**(*Style*, *Variant*, *Degree*)

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

Style Required [MsoGradientStyle](#). The gradient style.

MsoGradientStyle can be one of these MsoGradientStyle constants.

msoGradientDiagonalDown

msoGradientDiagonalUp

msoGradientFromCenter

msoGradientFromCorner

msoGradientFromTitle Used only in Microsoft PowerPoint.

msoGradientHorizontal

msoGradientMixed

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 **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 document.

```
With ActiveDocument.Shapes.AddShape(msoShapeRectangle, _  
    90, 90, 90, 80).Fill  
    .ForeColor.RGB = RGB(0, 128, 128)  
    .OneColorGradient msoGradientHorizontal, 1, 1  
End With
```



OnTime Method

Starts a background timer that runs a macro on the specified date and at the specified time.

expression.**OnTime**(*When*, *Name*, *Tolerance*)

expression Required. An expression that returns an **Application** object.

When Required **Variant**. The time at which the macro is to be run. Can be a string that specifies a time (for example, "4:30 pm" or "16:30"), or it can be a serial number returned by a function such as **TimeValue** or **TimeSerial** (for example, TimeValue("2:30 pm") or TimeSerial(14, 30, 00)). You can also include the date (for example, "6/30 4:15 pm" or TimeValue("6/30 4:15 pm")).

Use the sum of the return values of the **Now** function and either the **TimeValue** or **TimeSerial** function to set a timer to run a macro a specified amount of time after the statement is run. For example, use Now+TimeValue("00:05:30") to run a macro 5 minutes and 30 seconds after the statement is run.

Name Required **String**. The name of the macro to be run. Use the complete macro path to ensure that the correct macro is run (for example, "Project.Module1.Macro1"). For the macro to run, the document or template must be available both when the **OnTime** instruction is run and when the time specified by **When** arrives. For this reason, it's best to store the macro in Normal.dot or another global template that's loaded automatically.

Tolerance Optional **Variant**. The maximum time (in seconds) that can elapse before a macro that wasn't run at the time specified by **When** is canceled. Macros may not always run at the specified time. For example, if a sort operation is under way or a dialog box is being displayed, the macro will be delayed until Word has completed the task. If this argument is 0 (zero) or omitted, the macro is run regardless of how much time has elapsed since the time specified by **When**.

Remarks

Word can maintain only one background timer set by **OnTime**. If you start another timer before an existing timer runs, the existing timer is canceled.

Example

This example runs the macro named "Macro1" in the current module at 3:55 P.M.

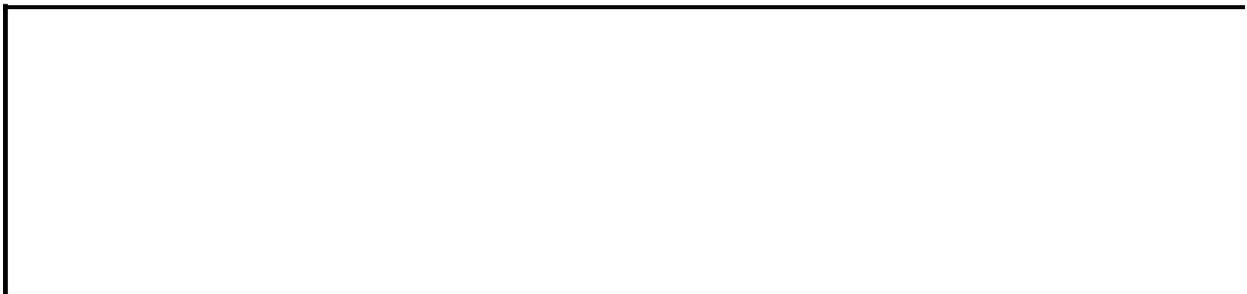
```
Application.OnTime When:="15:55:00", Name:="Macro1"
```

This example runs the macro named "Macro1" 15 seconds from the time the example is run. The macro name includes the project and module name.

```
Application.OnTime When:=Now + TimeValue("00:00:15"), _  
    Name:="Project1.Module1.Macro1"
```

This example runs the macro named "Start" at 1:30 P.M. The macro name includes the project and module name.

```
Application.OnTime When:=TimeValue("1:30 pm"), _  
    Name:="VBAProj.Module1.Start"
```



↳ [Show All](#)

Open Method

► [Open method as it applies to the **Documents** object.](#)

Opens the specified document and adds it to the [Documents](#) collection. Returns a [Document](#) object.

expression.Open(FileName, ConfirmConversions, ReadOnly, AddToRecentFiles, PasswordDocument, PasswordTemplate, Revert, WritePasswordDocument, WritePasswordTemplate, Format, Encoding, Visible, OpenConflictDocument, OpenAndRepair, DocumentDirection, NoEncodingDialog)

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

FileName Required **Variant**. The name of the document (paths are accepted).

ConfirmConversions Optional **Variant**. **True** to display the **Convert File** dialog box if the file isn't in Microsoft Word format.

ReadOnly Optional **Variant**. **True** to open the document as read-only. **Note** This argument doesn't override the read-only recommended setting on a saved document. For example, if a document has been saved with read-only recommended turned on, setting the **ReadOnly** argument to **False** will not cause the file to be opened as read/write.

AddToRecentFiles Optional **Variant**. **True** to add the file name to the list of recently used files at the bottom of the **File** menu.

PasswordDocument Optional **Variant**. The password for opening the document.

PasswordTemplate Optional **Variant**. The password for opening the template.

Revert Optional **Variant**. Controls what happens if **FileName** is the name of an open document. **True** to discard any unsaved changes to the open document and

reopen the file. **False** to activate the open document.

WritePasswordDocument Optional **VARIANT**. The password for saving changes to the document.

WritePasswordTemplate Optional **VARIANT**. The password for saving changes to the template.

Format Optional **VARIANT**. The file converter to be used to open the document. Can be one of the following [WdOpenFormat](#) constants.

WdOpenFormat can be one of these WdOpenFormat constants.

wdOpenFormatAllWord

wdOpenFormatAuto The default value.

wdOpenFormatDocument

wdOpenFormatEncodedText

wdOpenFormatRTF

wdOpenFormatTemplate

wdOpenFormatText

wdOpenFormatUnicodeText

wdOpenFormatWebPages

To specify an external file format, apply the [OpenFormat](#) property to a **FileConverter** object to determine the value to use with this argument.

Encoding Optional **VARIANT**. The document encoding (code page or character set) to be used by Microsoft Word when you view the saved document. Can be any valid **MsoEncoding** constant. For the list of valid **MsoEncoding** constants, see the Object Browser in the Visual Basic Editor. The default value is the system code page.

Visible Optional **VARIANT**. **True** if the document is opened in a visible window. The default value is **True**.

OpenConflictDocument Optional **VARIANT**. Specifies whether to open the conflict file for a document with an offline conflict.

OpenAndRepair Optional **VARIANT**. **True** to repair the document to prevent

document corruption.

DocumentDirection Optional [WdDocumentDirection](#). Indicates the horizontal flow of text in a document.

WdDocumentDirection can be one of these WdDocumentDirection constants.

wdLeftToRight *default*

wdRightToLeft

NoEncodingDialog Optional **Variant**. **True** to skip displaying the Encoding dialog box that Word displays if the text encoding cannot be recognized. The default value is **False**.

▶ [Open method as it applies to the **OLEFormat** object.](#)

Opens the specified object.

expression.**Open**

expression Required. An expression that returns an [OLEFormat](#) object.

▶ [Open method as it applies to the **RecentFile**, **Subdocument**, and **Version** objects.](#)

Opens the specified object. Returns a [Document](#) object representing the opened object.

expression.**Open**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Documents** object.](#)

This example opens MyDoc.doc as a read-only document.

```
Sub OpenDoc()  
    Documents.Open FileName:="C:\MyFiles\MyDoc.doc", ReadOnly:=True  
End Sub
```

This example opens Test.wp using the WordPerfect 6.x file converter.

```
Sub OpenDoc2()  
    Dim fmt As Variant  
    fmt = Application.FileConverters("WordPerfect6x").OpenFormat  
    Documents.Open FileName:="C:\MyFiles\Test.wp", Format:=fmt  
End Sub
```

▶ [As it applies to the **RecentFiles** object.](#)

This example opens each document in the **RecentFiles** collection.

```
Sub OpenRecentFiles()  
    Dim rFile As RecentFile  
    For Each rFile In RecentFiles  
        rFile.Open  
    Next rFile  
End Sub
```

▶ [As it applies to the **Version** object.](#)

This example opens the most recent version of Report.doc.

```
Sub OpenVersion()  
    Dim mydoc As Document  
    Set mydoc = Documents.Open("C:\MyFiles\Report.doc")  
    If mydoc.Versions.Count > 0 Then  
        mydoc.Versions(mydoc.Versions.Count).Open  
    Else  
        MsgBox "There are no saved versions for this document."  
    End If  
End Sub
```



OpenAsDocument Method

-
Opens the specified template as a document and returns a **Document** object.

Note Opening a template as a document allows the user to edit the contents of the template. This may be necessary if a property or method (the **Styles** property, for example) isn't available from the **Template** object.

expression.**OpenAsDocument()**

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

Example

This example opens the template attached to the active document, displays a message box if the template contains anything more than a single paragraph mark, and then closes the template.

```
Dim docNew As Document

Set docNew = ActiveDocument.AttachedTemplate.OpenAsDocument

If docNew.Content.Text <> Chr(13) Then
    MsgBox "Template is not empty"
Else
    MsgBox "Template is empty"
End If
docNew.Close SaveChanges:=wdDoNotSaveChanges
```

This example saves a copy of the Normal template as "Backup.dot."

```
Dim docNew As Document

Set docNew = NormalTemplate.OpenAsDocument

With docNew
    .SaveAs FileName:="Backup.dot"
    .Close SaveChanges:=wdDoNotSaveChanges
End With
```

This example changes the formatting of the Heading 1 style in the template attached to the active document. The **UpdateStyles** method updates the styles in the active document.

```
Dim docNew As Document

Set docNew = ActiveDocument.AttachedTemplate.OpenAsDocument

With docNew.Styles(wdStyleHeading1).Font
    .Name = "Arial"
    .Size = 16
    .Bold = False
End With
docNew.Close SaveChanges:=wdSaveChanges
ActiveDocument.UpdateStyles
```



↳ [Show All](#)

OpenDataSource Method

Attaches a data source to the specified document, which becomes a main document if it's not one already.

expression.OpenDataSource(Name, Format, ConfirmConversions, ReadOnly, LinkToSource, AddToRecentFiles, PasswordDocument, PasswordTemplate, Revert, WritePasswordDocument, WritePasswordTemplate, Connection, SQLStatement, SQLStatement1, OpenExclusive)

expression Required. An expression that returns a [MailMerge](#) object.

Name Required **String**. The data source file name. You can specify a Microsoft Query (.qry) file instead of specifying a data source, a connection string, and a query string.

Format Optional **Variant**. The file converter used to open the document. Can be one of the [WdOpenFormat](#) constants. To specify an external file format, use the [OpenFormat](#) property with the [FileConverter](#) object to determine the value to use with this argument.

WdOpenFormat can be one of these WdOpenFormat constants.

wdOpenFormatAllWord

wdOpenFormatAuto Default.

wdOpenFormatDocument

wdOpenFormatEncodedText

wdOpenFormatRTF

wdOpenFormatTemplate

wdOpenFormatText

wdOpenFormatUnicodeText

wdOpenFormatWebPages

ConfirmConversions Optional **Variant**. **True** to display the **Convert File**

dialog box if the file isn't in Word format.

ReadOnly Optional **Variant**. **True** to open the data source on a read-only basis.

LinkToSource Optional **Variant**. **True** to perform the query specified by **Connection** and **SQLStatement** each time the main document is opened.

AddToRecentFiles Optional **Variant**. **True** to add the file name to the list of recently used files at the bottom of the **File** menu.

PasswordDocument Optional **Variant**. The password used to open the data source.

PasswordTemplate Optional **Variant**. The password used to open the template.

Revert Optional **Variant**. Controls what happens if **Name** is the file name of an open document. **True** to discard any unsaved changes to the open document and reopen the file; **False** to activate the open document.

WritePasswordDocument Optional **Variant**. The password used to save changes to the document.

WritePasswordTemplate Optional **Variant**. The password used to save changes to the template.

Connection Optional **Variant**. A range within which the query specified by **SQLStatement** is to be performed. How you specify the range depends on how data is retrieved. For example:

- When retrieving data through ODBC, you specify a connection string.
- When retrieving data from Microsoft Excel using dynamic data exchange (DDE), you specify a named range.
- When retrieving data from Microsoft Access, you specify the word "Table" or "Query" followed by the name of a table or query.

SQLStatement Optional **Variant**. Defines query options for retrieving data.

SQLStatement1 Optional **Variant**. If the query string is longer than 255 characters, **SQLStatement** specifies the first portion of the string, and **SQLStatement1** specifies the second portion.

OpenExclusive Optional **Variant**. **True** to open exclusively.

Remarks

To determine the ODBC connection and query strings, set query options manually, and use the [QueryString](#) property to return the connection string. The following table includes some commonly used SQL keywords.

Keyword	Description
DSN	The name of the ODBC data source
UID	The user logon ID
PWD	The user-specified password
DBQ	The database file name
FIL	The file type

Example

This example creates a new main document and attaches the Orders table from a Microsoft Access database named "Northwind.mdb."

```
Dim docNew As Document

Set docNew = Documents.Add

With docNew.MailMerge
    .MainDocumentType = wdFormLetters
    .OpenDataSource _
        Name:="C:\Program Files\Microsoft Office" & _
        "\Office\Samples\Northwind.mdb", _
        LinkToSource:=True, AddToRecentFiles:=False, _
        Connection:="TABLE Orders"
End With
```

This example creates a new main document and attaches the Microsoft Excel spreadsheet named "Names.xls." The **Connection** argument retrieves data from the range named "Sales."

```
Dim docNew As Document

Set docNew = Documents.Add

With docNew.MailMerge
    .MainDocumentType = wdCatalog
    .OpenDataSource Name:="C:\Documents\Names.xls", _
        ReadOnly:=True, _
        Connection:="Sales"
End With
```

This example uses ODBC to attach the Microsoft Access database named "Northwind.mdb" to the active document. The **SQLStatement** argument selects the records in the Customers table.

```
Dim strConnection As String

With ActiveDocument.MailMerge
    .MainDocumentType = wdFormLetters
    strConnection = "DSN=MS Access Databases;" _
        & "DBQ=C:\Northwind.mdb;" _
```

```
        & "FIL=RedISAM;"  
    .OpenDataSource Name:="C:\NorthWind.mdb", _  
        Connection:=strConnection, _  
        SQLStatement:="SELECT * FROM Customers"  
End With
```



↳ [Show All](#)

OpenHeaderSource Method

Attaches a mail merge header source to the specified document.

expression.**OpenHeaderSource**(*Name*, *Format*, *ConfirmConversions*, *ReadOnly*, *AddToRecentFiles*, *PasswordDocument*, *PasswordTemplate*, *Revert*, *WritePasswordDocument*, *WritePasswordTemplate*, *OpenExclusive*)

expression Required. An expression that returns a [MailMerge](#) object.

Name Required **String**. The file name of the header source.

Format Optional **Variant**. The file converter used to open the document. Can be one of the following [WdOpenFormat](#) constants. To specify an external file format, use the [OpenFormat](#) property with a [FileConverter](#) object to determine the value to use with this argument.

WdOpenFormat can be one of these WdOpenFormat constants.

wdOpenFormatAllWord

wdOpenFormatAuto Default.

wdOpenFormatDocument

wdOpenFormatEncodedText

wdOpenFormatRTF

wdOpenFormatTemplate

wdOpenFormatText

wdOpenFormatUnicodeText

wdOpenFormatWebPages

ConfirmConversions Optional **Variant**. **True** to display the **Convert File** dialog box if the file isn't in Word format.

ReadOnly Optional **Variant**. **True** to open the header source on a read-only basis.

AddToRecentFiles Optional **Variant**. **True** to add the file name to the list of recently used files at the bottom of the **File** menu.

PasswordDocument Optional **Variant**. The password required to open the header source document.

PasswordTemplate Optional **Variant**. The password required to open the header source template.

Revert Optional **Variant**. Controls what happens if **Name** is the file name of an open document. **True** to discard any unsaved changes to the open document and reopen the file; **False** to activate the open document.

WritePasswordDocument Optional **Variant**. The password required to save changes to the document data source.

WritePasswordTemplate Optional **Variant**. The password required to save changes to the template data source.

OpenExclusive Optional **Variant**. **True** to open exclusively.

Remarks

When a header source is attached, the first record in the header source is used in place of the header record in the data source.

Example

This example sets the active document as a main document for form letters, and then it attaches the header source named "Header.doc" and the data document named "Names.doc."

```
With ActiveDocument.MailMerge
    .MainDocumentType = wdFormLetters
    .OpenHeaderSource Name:="C:\Documents\Header.doc", _
        Revert:=False, AddToRecentFiles:=False
    .OpenDataSource Name:="C:\Documents\Names.doc"
End With
```



OpenOrCloseUp Method

-

If spacing before the specified paragraphs is 0 (zero), this method sets spacing to 12 points. If spacing before the paragraphs is greater than 0 (zero), this method sets spacing to 0 (zero).

expression.**OpenOrCloseUp**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Example

This example toggles the formatting of the first paragraph in the active document to either add 12 points of space before the paragraph or leave no space before it.

`ActiveDocument.Paragraphs(1).OpenOrCloseUp`



OpenUp Method

-
Sets spacing before the specified paragraphs to 12 points.

expression.**OpenUp**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Remarks

The following two statements are equivalent:

```
ActiveDocument.Paragraphs(1).OpenUp  
ActiveDocument.Paragraphs(1).SpaceBefore = 12
```

Example

This example changes the formatting of the second paragraph in the active document to leave 12 points of space before the paragraph.

ActiveDocument.Paragraphs(2).OpenUp



Options Method

-
Displays the **Envelope Options** dialog box.

expression.**Options**

expression Required. An expression that returns an [Envelope](#) object.

Remarks

The **Options** method works only if the document is the main document of an envelope mail merge.

Example

This example checks that the active document is an envelope mail merge main document, and if it is, displays the **Envelope Options** dialog box.

```
Sub EnvelopeOptions()  
    If ThisDocument.MailMerge.MainDocumentType = wdEnvelopes Then  
        ActiveDocument.Envelope.Options  
    End If  
End Sub
```



OrganizerCopy Method

-

Copies the specified AutoText entry, toolbar, style, or macro project item from the source document or template to the destination document or template.

expression.**OrganizerCopy**(*Source*, *Destination*, *Name*, *Object*)

expression Required. An expression that returns an **Application** object.

Source Required **String**. The document or template file name that contains the item you want to copy.

Destination Required **String**. The document or template file name to which you want to copy an item.

Name Required **String**. The name of the AutoText entry, toolbar, style, or macro you want to copy.

Object Required [WdOrganizerObject](#). The kind of item you want to copy.

WdOrganizerObject can be one of these WdOrganizerObject constants.

wdOrganizerObjectAutoText

wdOrganizerObjectCommandBars

wdOrganizerObjectProjectItems

wdOrganizerObjectStyles

Example

This example copies all the AutoText entries in the template attached to the active document to the Normal template.

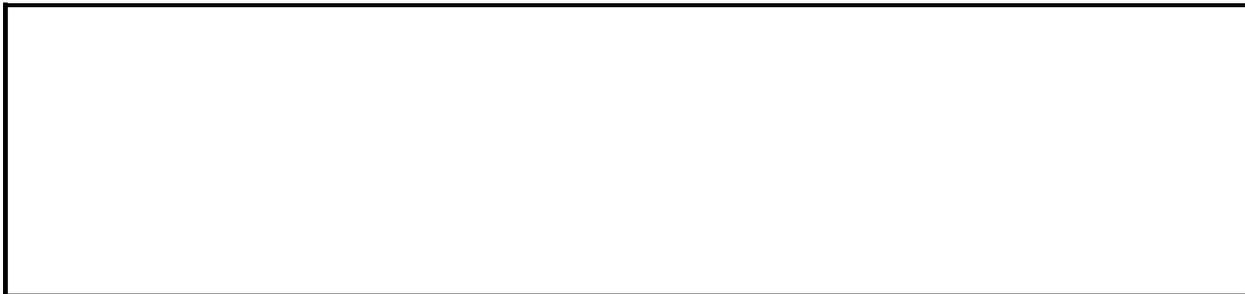
```
Dim atEntry As AutoTextEntry

For Each atEntry In _
    ActiveDocument.AttachedTemplate.AutoTextEntries
    Application.OrganizerCopy _
        Source:=ActiveDocument.AttachedTemplate.FullName, _
        Destination:=NormalTemplate.FullName, Name:=atEntry.Name, _
        Object:=wdOrganizerObjectAutoText
Next atEntry
```

If the style named "SubText" exists in the active document, this example copies the style to C:\Templates\Template1.dot.

```
Dim styleLoop As Style

For Each styleLoop In ActiveDocument.Styles
    If styleLoop = "SubText" Then
        Application.OrganizerCopy Source:=ActiveDocument.Name, _
            Destination:="C:\Templates\Template1.dot", _
            Name:="SubText", _
            Object:=wdOrganizerObjectStyles
    End If
Next styleLoop
```



OrganizerDelete Method

-

Deletes the specified style, AutoText entry, toolbar, or macro project item from a document or template.

expression.**OrganizerDelete**(*Source*, *Name*, *Object*)

expression Required. An expression that returns an **Application** object.

Source Required **String**. The file name of the document or template that contains the item you want to delete.

Name Required **String**. The name of the style, AutoText entry, toolbar, or macro you want to delete.

Object Required [WdOrganizerObject](#). The kind of item you want to copy.

WdOrganizerObject can be one of these WdOrganizerObject constants.

wdOrganizerObjectAutoText

wdOrganizerObjectCommandBars

wdOrganizerObjectProjectItems

wdOrganizerObjectStyles

Example

This example deletes the toolbar named "Custom 1" from the Normal template.

```
Dim cbLoop As CommandBar

For Each cbLoop In CommandBars
    If cbLoop.Name = "Custom 1" Then
        Application.OrganizerDelete Source:=NormalTemplate.Name, _
            Name:="Custom 1", _
            Object:=wdOrganizerObjectCommandBars
    End If
Next cbLoop
```

This example prompts the user to delete each AutoText entry in the template attached to the active document. If the user clicks the Yes button, the AutoText entries are deleted.

```
Dim atEntry As AutoTextEntry
Dim intResponse As Integer

For Each atEntry In _
    ActiveDocument.AttachedTemplate.AutoTextEntries
    intResponse = _
        MsgBox("Do you want to delete the " & atEntry.Name _
            & " AutoText entry?", vbYesNoCancel)
    If intResponse = vbYes Then
        With ActiveDocument.AttachedTemplate
            Application.OrganizerDelete _
                Source:= .Path & "\" & .Name, _
                Name:=atEntry.Name, _
                Object:=wdOrganizerObjectAutoText
        End With
    ElseIf intResponse = vbCancel Then
        Exit For
    End If
Next atEntry
```



OrganizerRename Method

Renames the specified style, AutoText entry, toolbar, or macro project item in a document or template.

expression.**OrganizerRename**(*Source*, *Name*, *NewName*, *Object*)

expression Required. An expression that returns an **Application** object.

Source Required **String**. The file name of the document or template that contains the item you want to rename.

Name Required **String**. The name of the style, AutoText entry, toolbar, or macro you want to rename.

NewName Required **String**. The new name for the item.

Object Required [WdOrganizerObject](#). The kind of item you want to copy.

WdOrganizerObject can be one of these WdOrganizerObject constants.

wdOrganizerObjectAutoText

wdOrganizerObjectCommandBars

wdOrganizerObjectProjectItems

wdOrganizerObjectStyles

Example

This example changes the name of the style named "SubText" in the active document to "SubText2."

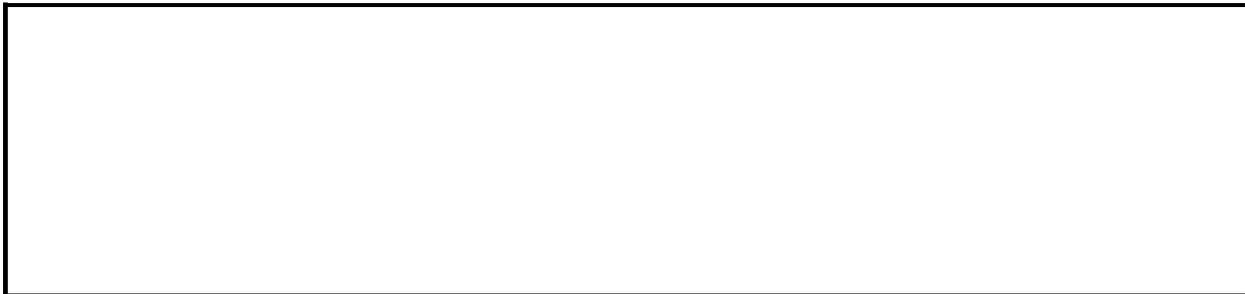
```
Dim styleLoop as Style

For Each styleLoop In ActiveDocument.Styles
    If styleLoop.NameLocal = "SubText" Then
        Application.OrganizerRename _
            Source:=ActiveDocument.Name, Name:="SubText", _
            NewName:="SubText2", _
            Object:=wdOrganizerObjectStyles
    End If
Next styleLoop
```

This example changes the name of the macro module named "Module1" in the attached template to "Macros1."

```
Dim dotTemp As Template

dotTemp = ActiveDocument.AttachedTemplate.Name
Application.OrganizerRename Source:=dotTemp, Name:="Module1", _
    NewName:="Macros1", Object:=wdOrganizerObjectProjectItems
```



Outdent Method

-
Removes one level of indent for one or more paragraphs.

Note Using this method is equivalent to clicking the **Decrease Indent** button on the **Formatting** toolbar.

expression.**Outdent**

expression Required. An expression that returns a **Paragraph** or **Paragraphs** object.

Example

This example indents all the paragraphs in the active document twice, and then it removes one level of the indent for the first paragraph.

```
With ActiveDocument.Paragraphs
    .Indent
    .Indent
End With
ActiveDocument.Paragraphs(1).Outdent
```



OutlineDemote Method

-

Applies the next heading level style (Heading 1 through Heading 8) to the specified paragraph or paragraphs. For example, if a paragraph is formatted with the Heading 2 style, this method demotes the paragraph by changing the style to Heading 3.

expression.**OutlineDemote**

expression Required. An expression that returns a **Paragraph** or **Paragraphs** object.

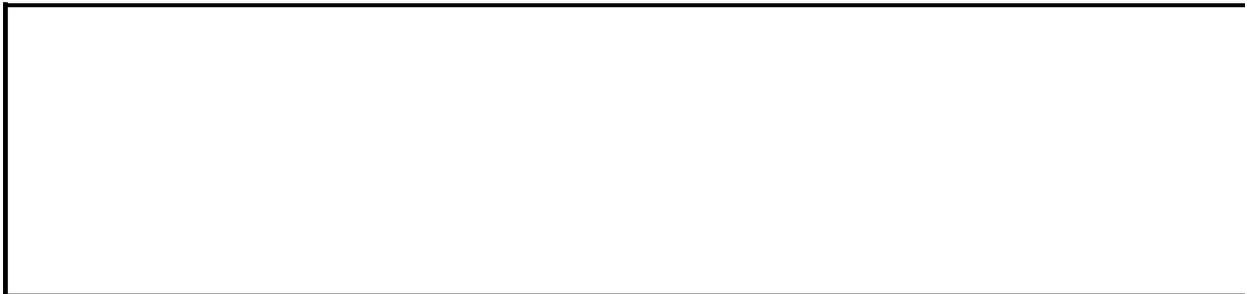
Example

This example demotes the selected paragraphs.

```
Selection.Paragraphs.OutlineDemote
```

This example demotes the third paragraph in the active document.

```
ActiveDocument.Paragraphs(3).OutlineDemote
```



OutlineDemoteToBody Method

-
Demotes the specified paragraph or paragraphs to body text by applying the Normal style.

expression.**OutlineDemoteToBody**

expression Required. An expression that returns a **Paragraph** or **Paragraphs** object.

Example

This example demotes the selected paragraphs to body text by applying the Normal style.

```
Selection.Paragraphs.OutlineDemoteToBody
```

This example switches the active window to outline view and demotes the first paragraph in the selection to body text.

```
ActiveDocument.ActiveWindow.View.Type = wdOutlineView  
Selection.Paragraphs(1).OutlineDemoteToBody
```



OutlinePromote Method

-

Applies the previous heading level style (Heading 1 through Heading 8) to the specified paragraph or paragraphs. For example, if a paragraph is formatted with the Heading 2 style, this method promotes the paragraph by changing the style to Heading 1.

expression.**OutlinePromote**

expression Required. An expression that returns a **Paragraph** or **Paragraphs** object.

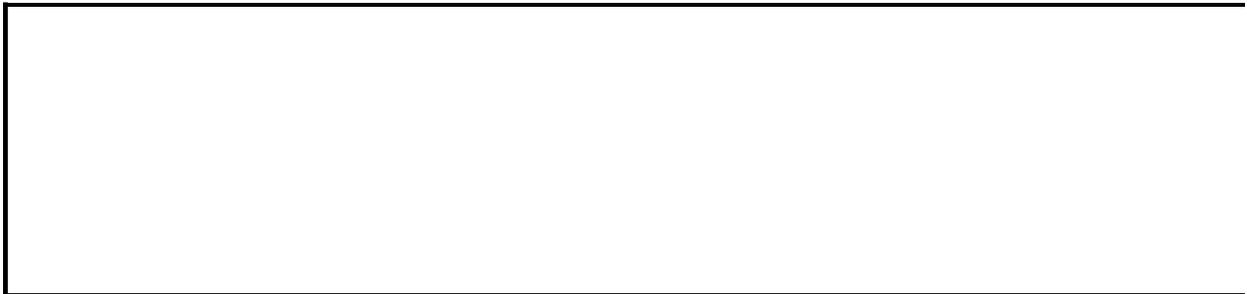
Example

This example promotes the selected paragraphs.

```
Selection.Paragraphs.OutlinePromote
```

This example switches the active window to outline view and promotes the first paragraph in the active document.

```
ActiveDocument.ActiveWindow.View.Type = wdOutlineView  
ActiveDocument.Paragraphs(1).OutlinePromote
```



PageScroll Method

-
Scrolls through the specified pane or window page by page.

expression.**PageScroll**(*Down*, *Up*)

expression Required. An expression that returns a **Window** or **Pane** object.

Down Optional **Variant**. The number of pages to be scrolled down. If this argument is omitted, this value is assumed to be 1.

Up Optional **Variant**. The number of pages to be scrolled up.

Remarks

The **PageScroll** method is available only if you're in print layout view or web layout view. This method doesn't affect the position of the insertion point.

If **Down** and **Up** are both specified, the window is scrolled by the difference of the arguments. For example, if **Down** is 2 and **Up** is 4, the window is scrolled up two pages.

Example

This example scrolls down three pages in the active window.

```
ActiveDocument.ActiveWindow.View.Type = wdPrintView  
ActiveDocument.ActiveWindow.PageScroll Down:=3
```

This example scrolls up one page in the active pane.

```
ActiveDocument.ActiveWindow.View.Type = wdPrintView  
ActiveDocument.ActiveWindow.ActivePane.PageScroll Up:=1
```

This example scrolls down one page in the active window.

```
ActiveDocument.ActiveWindow.View.Type = wdPrintView  
ActiveDocument.ActiveWindow.PageScroll
```



Paste Method

-

Inserts the contents of the Clipboard at the specified range or selection. If you don't want to replace the contents of the range or selection, use the **Collapse** method before using this method.

expression.**Paste**

expression Required. An expression that returns a **Range** or **Selection** object.

Remarks

When this method is used with a range object, the range expands to include the contents of the Clipboard. When this method is used with a selection object, the selection doesn't expand to include the Clipboard contents; instead, the selection is positioned after the pasted Clipboard contents.

Example

This example copies and pastes the first table in the active document into a new document.

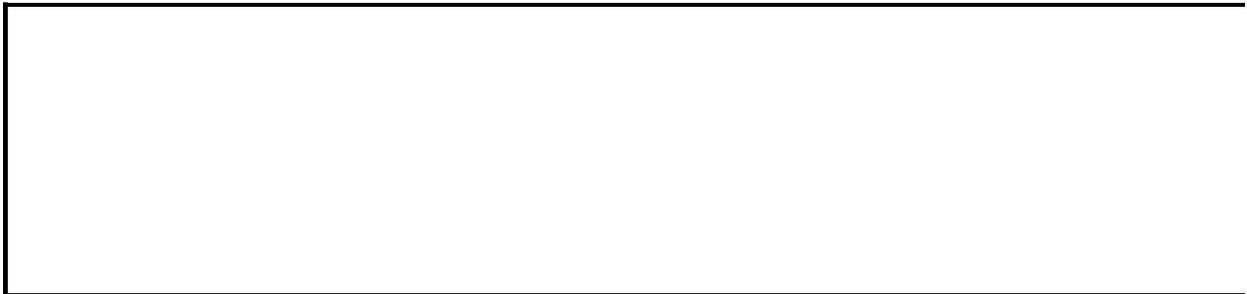
```
If ActiveDocument.Tables.Count >= 1 Then
    ActiveDocument.Tables(1).Range.Copy
    Documents.Add.Content.Paste
End If
```

This example copies the first paragraph in the document and pastes it at the insertion point.

```
ActiveDocument.Paragraphs(1).Range.Copy
Selection.Collapse Direction:=wdCollapseStart
Selection.Paste
```

This example copies the selection and pastes it at the end of the document.

```
If Selection.Type <> wdSelectionIP Then
    Selection.Copy
    Set Range2 = ActiveDocument.Content
    Range2.Collapse Direction:=wdCollapseEnd
    Range2.Paste
End If
```



↳ [Show All](#)

PasteAndFormat Method

Pastes the selected table cells and formats them as specified.

expression.**PasteAndFormat**(*Type*)

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

Type Required [WdRecoveryType](#). The type of formatting to use when pasting the selected table cells.

WdRecoveryType can be one of these WdRecoveryType constants.

wdChart Pastes a Microsoft Excel chart as an embedded OLE object.

wdChartLinked Pastes an Excel chart and links it to the original Excel spreadsheet.

wdChartPicture Pastes an Excel chart as a picture.

wdFormatOriginalFormatting Preserves original formatting of the pasted material.

wdFormatPlainText Pastes as plain, unformatted text.

wdFormatSurroundingFormattingWithEmphasis Matches the formatting of the pasted text to the formatting of surrounding text.

wdListCombineWithExistingList Merges a pasted list with neighboring lists.

wdListContinueNumbering Continues numbering of a pasted list from the list in the document.

wdListRestartNumbering Restarts numbering of a pasted list.

wdSingleCellTable Pastes a single cell table as a separate table.

wdSingleCellText Pastes a single cell as text.

wdTableAppendTable Merges pasted cells into an existing table by inserting the pasted rows between the selected rows.

wdTableInsertAsRows Inserts a pasted table as rows between two rows in the target table.

wdTableOriginalFormatting Pastes an appended table without merging table styles.

wdTableOverwriteCells Pastes table cells and overwrites existing table cells.

Example

This example pastes a selected Microsoft Excel chart as a picture. This example assumes that the Clipboard contains an Excel chart.

```
Sub PasteChart()  
    Selection.PasteAndFormat Type:=wdChartPicture  
End Sub
```



PasteAppendTable Method

-

Merges pasted cells into an existing table by inserting the pasted rows between the selected rows. No cells are overwritten.

expression.**PasteAppendTable**

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

Example

This example pastes table cells by inserting rows into the current table at the insertion point. This example assumes that the Clipboard contains a collection of table cells.

```
Sub PasteAppend  
    Selection.PasteAppendTable  
End Sub
```



PasteAsNestedTable Method

-
Pastes a cell or group of cells as a nested table into the selected range.

expression.**PasteAsNestedTable**

expression Required. An expression that returns a **Range** or **Selection** object.

Remarks

You can use **PasteAsNestedTable** only if the Clipboard contains a cell or group of cells and the selected range is a cell or group of cells in the current document.

Example

This example pastes the contents of the Clipboard into the third cell of the first table in the active document.

```
ActiveDocument.Tables(1).Rows(1).Cells(3).Range _  
    .PasteAsNestedTable
```



PasteExcelTable Method

Pastes and formats a Microsoft Excel table.

expression.**PasteExcelTable**(*LinkedToExcel*, *WordFormatting*, *RTF*)

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

LinkedToExcel Required **Boolean**. **True** links the pasted table to the original Excel file so that changes made to the Excel file are reflected in Microsoft Word.

WordFormatting Required **Boolean**. **True** formats the table using the formatting in the Word document. **False** formats the table according to the original Excel file.

RTF Required **Boolean**. **True** pastes the Excel table using Rich Text Format (RTF). **False** pastes the Excel table as HTML.

Example

This example pastes an Excel table into the active document. The parameters specify that the pasted table is linked to the Excel file, retains the original Excel formatting, and is pasted as RTF. This example assumes that the Clipboard contains an Excel table.

```
Sub PasteExcelFormatted()  
    Selection.PasteExcelTable _  
        LinkedToExcel:=True, _  
        WordFormatting:=False, _  
        RTF:=True  
End Sub
```



PasteFormat Method

-

Applies formatting copied with the [CopyFormat](#) method to the selection. If a paragraph mark was selected when the **CopyFormat** method was used, Word applies paragraph formatting in addition to character formatting.

expression.**PasteFormat**

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

Example

This example copies the paragraph and character formatting from the first paragraph in the selection to the next paragraph in the selection.

```
With Selection
    .Paragraphs(1).Range.Select
    .CopyFormat
    .Paragraphs(1).Next.Range.Select
    .PasteFormat
End With
```

This example collapses the selection and copies the character formatting to the next word.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .CopyFormat
    .Next(Unit:=wdWord, Count:=1).Select
    .PasteFormat
End With
```



PasteSpecial Method

Inserts the contents of the Clipboard. Unlike with the **Paste** method, with **PasteSpecial** you can control the format of the pasted information and (optionally) establish a link to the source file (for example, a Microsoft Excel worksheet).

Note If you don't want to replace the contents of the specified range or selection, use the **Collapse** method before you use this method. When you use this method, the range or selection doesn't expand to include the contents of the Clipboard.

expression.PasteSpecial(IconIndex, Link, Placement, DisplayAsIcon, DataType, IconFileName, IconLabel)

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

IconIndex Optional **Variant**. If **DisplayAsIcon** is **True**, this argument is a number that corresponds to the icon you want to use in the program file specified by **IconFilename**. Icons appear in the **Change Icon** dialog box (**Insert** menu, **Object** command, **Create New** tab): 0 (zero) corresponds to the first icon, 1 corresponds to the second icon, and so on. If this argument is omitted, the first (default) icon is used.

Link Optional **Variant**. **True** to create a link to the source file of the Clipboard contents. The default value is **False**.

Placement Optional **Variant**. Can be either of the following **WdOLEPlacement** constants: **wdFloatOverText** or **wdInLine**. The default value is **wdInLine**.

DisplayAsIcon Optional **Variant**. Optional **Variant**. **True** to display the link as an icon. The default value is **False**.

DataType Optional **Variant**. A format for the Clipboard contents when they're inserted into the document. [WdPastDataType](#).

Can be one of the following **WdPasteDataType** constants

wdPasteBitmap

wdPasteDeviceIndependentBitmap

wdPasteEnhancedMetafile

wdPasteHTML

wdPasteHyperlink

wdPasteMetafilePicture

wdPasteOLEObject

wdPasteRTF

wdPasteShape

wdPasteText

The default format varies, depending on the contents of the Clipboard.

IconFileName Optional **Variant**. If ***DisplayAsIcon*** is **True**, this argument is the path and file name for the file in which the icon to be displayed is stored.

IconLabel Optional **Variant**. If ***DisplayAsIcon*** is **True**, this argument is the text that appears below the icon.

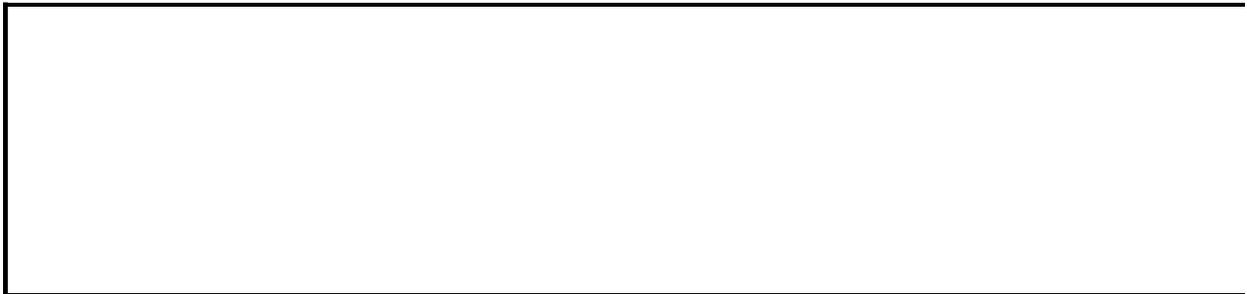
Example

This example inserts the Clipboard contents at the insertion point as unformatted text.

```
Selection.Collapse Direction:=wdCollapseStart  
Selection.Range.PasteSpecial DataType:=wdPasteText
```

This example copies the selected text and pastes it into a new document as a hyperlink. The source document must first be saved for this example to work.

```
If Selection.Type = wdSelectionNormal Then  
    Selection.Copy  
    Documents.Add.Content.PasteSpecial Link:=True, _  
        DataType:=wdPasteHyperlink  
End If
```



↳ [Show All](#)

Patterned Method

Sets the specified fill to a pattern.

expression.**Patterned**(*Pattern*)

expression Required. An expression that returns a [FillFormat](#) object.

Pattern Required [MsoPatternType](#). The pattern to be used for the specified fill.

MsoPatternType can be one of these MsoPatternType constants.

msoPattern10Percent

msoPattern25Percent

msoPattern40Percent

msoPattern5Percent

msoPattern70Percent

msoPattern80Percent

msoPatternDarkDownwardDiagonal

msoPatternDarkUpwardDiagonal

msoPatternDashedDownwardDiagonal

msoPattern20Percent

msoPattern30Percent

msoPattern50Percent

msoPattern60Percent

msoPattern75Percent

msoPattern90Percent

msoPatternDarkHorizontal

msoPatternDarkVertical

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

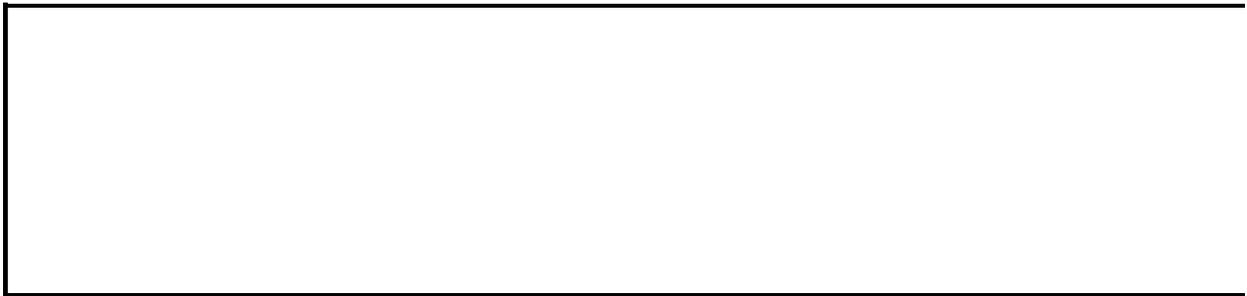
Remarks

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 document.

```
Sub FillPattern()  
  With ActiveDocument.Shapes.AddShape _  
    (msoShapeOval, 60, 60, 80, 40).Fill  
    .ForeColor.RGB = RGB(128, 0, 0)  
    .BackColor.RGB = RGB(0, 0, 255)  
    .Patterned msoPatternDarkVertical  
  End With  
End Sub
```



↳ [Show All](#)

PhoneticGuide Method

Adds phonetic guides to the specified range.

expression.**PhoneticGuide**(*Text*, *Alignment*, *Raise*, *FontSize*, *FontName*)

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

Text Required **String**. The phonetic text to add.

Alignment Optional [WdPhoneticGuideAlignmentType](#). The alignment of the added phonetic text.

WdPhoneticGuideAlignmentType can be one of these WdPhoneticGuideAlignmentType constants.

wdPhoneticGuideAlignmentCenter Microsoft Word centers phonetic text over the specified range. This is the default value.

wdPhoneticGuideAlignmentLeft Word left-aligns phonetic text with the specified range.

wdPhoneticGuideAlignmentOneTwoOne Word adjusts the inside and outside spacing of the phonetic text in a 1:2:1 ratio.

wdPhoneticGuideAlignmentRight Word right-aligns phonetic text with the specified range.

wdPhoneticGuideAlignmentRightSuperscript Word right-aligns superscript text with the specified range.

wdPhoneticGuideAlignmentZeroOneZero Word adjusts the inside and outside spacing of the phonetic text in a 0:1:0 ratio.

Raise Optional **Long**. 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, Microsoft Word automatically sets the phonetic text at an optimum distance above the specified range.

FontSize Optional **Long**. The font size to use for the phonetic text. If no value is specified, Word uses a font size 50% smaller than the text in the specified range.

FontName Optional **String**. The name of the font to use for the phonetic text. If no value is specified, Word uses the same font as the text in the specified range.

Remarks

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example adds a phonetic guide to the selected phrase "tres chic."

```
Selection.Range.PhoneticGuide Text:="tray sheek", _  
    Alignment:= wdPhoneticGuideAlignmentCenter, _  
    Raise:=11, FontSize:=7
```



PicasToPoints Method

-
Converts a measurement from picas to points (1 pica = 12 points). Returns the converted measurement as a **Single**.

expression.**PicasToPoints**(*Picas*)

expression Optional. An expression that returns an **Application** object.

Picas Required **Single**. The pica value to be converted to points.

Example

This example adds line numbers to the active document and sets the distance between the line numbers and the document text to 4 picas.

```
With ActiveDocument.PageSetup.LineNumbering
    .Active = True
    .DistanceFromText = PicasToPoints(4)
End With
```

This example sets the first-line indent for the selected paragraphs to 3 picas.

```
Selection.ParagraphFormat.FirstLineIndent = PicasToPoints(3)
```



PickUp Method

-

Copies the formatting of the specified shape. Use the [Apply](#) method to apply the copied formatting to another shape.

expression.**PickUp**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Example

This example copies the formatting of shape one on myDocument and then applies the copied formatting to shape two.

```
Set myDocument = ActiveDocument
With myDocument
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```



PixelsToPoints Method

-
Converts a measurement from pixels to points. Returns the converted measurement as a **Single**.

expression.**PixelsToPoints**(*Pixels*, *fVertical*)

expression Required. An expression that returns an **Application** object.

Pixels Required **Single**. The pixel value to be converted to points.

fVertical Optional **Variant**. **True** to convert vertical pixels; **False** to convert horizontal pixels.

Example

This example displays the height and width in points of an object measured in pixels.

```
MsgBox "320x240 pixels is equivalent to " _  
    & PixelsToPoints(320, False) & "x" _  
    & PixelsToPoints(240, True) _  
    & " points on this display."
```



PointsToCentimeters Method

-
Converts a measurement from points to centimeters (1 centimeter = 28.35 points). Returns the converted measurement as a **Single**.

expression.**PointsToCentimeters**(*Points*)

expression Optional. An expression that returns an **Application** object.

Points Required **Single**. The measurement, in points.

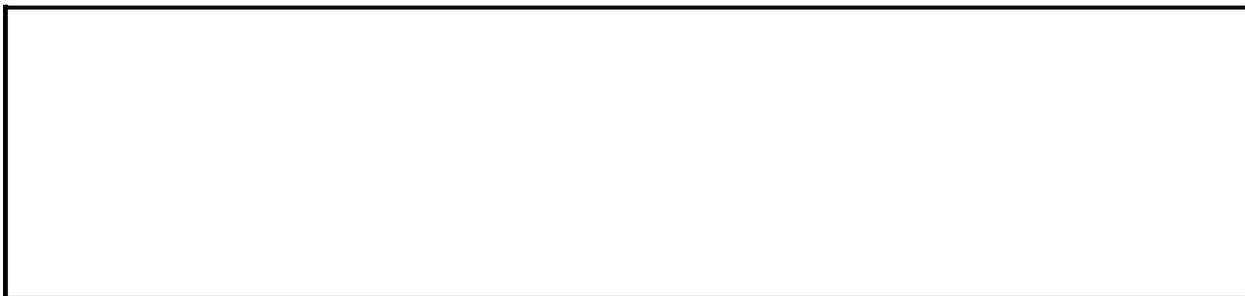
Example

This example converts a measurement of 30 points to the corresponding number of centimeters.

```
MsgBox PointsToCentimeters(30) & " centimeters"
```

This example converts the value of the variable `sngData` (a measurement in points) to centimeters, inches, lines, millimeters, or picas, depending on the value of the variable `intUnit` (a value from 1 through 5 that indicates the resulting unit of measurement).

```
Function ConvertPoints(ByVal intUnit As Integer, _  
    sngData As Single) As Single  
  
    Select Case intUnit  
        Case 1  
            ConvertPoints = PointsToCentimeters(sngData)  
        Case 2  
            ConvertPoints = PointsToInches(sngData)  
        Case 3  
            ConvertPoints = PointsToLines(sngData)  
        Case 4  
            ConvertPoints = PointsToMillimeters(sngData)  
        Case 5  
            ConvertPoints = PointsToPicas(sngData)  
        Case Else  
            Error 5  
    End Select  
  
End Function
```



PointsToInches Method

-
Converts a measurement from points to inches (1 inch = 72 points). Returns the converted measurement as a **Single**.

expression.**PointsToInches**(*Points*)

expression Optional. An expression that returns an **Application** object.

Points Required **Single**. The measurement, in points.

Example

This example converts the measurement of the top margin for the active document to inches and displays the result in a message box.

```
MsgBox PointsToInches(ActiveDocument.Sections(1) _  
    .PageSetup.TopMargin)
```

This example converts the value of the variable `sngData` (a measurement in points) to centimeters, inches, lines, millimeters, or picas, depending on the value of the variable `intUnit` (a value from 1 through 5 that indicates the resulting unit of measurement).

```
Function ConvertPoints(ByVal intUnit As Integer, _  
    sngData As Single) As Single  
  
    Select Case intUnit  
        Case 1  
            ConvertPoints = PointsToCentimeters(sngData)  
        Case 2  
            ConvertPoints = PointsToInches(sngData)  
        Case 3  
            ConvertPoints = PointsToLines(sngData)  
        Case 4  
            ConvertPoints = PointsToMillimeters(sngData)  
        Case 5  
            ConvertPoints = PointsToPicas(sngData)  
        Case Else  
            Error 5  
    End Select  
  
End Function
```



PointsToLines Method

-
Converts a measurement from points to lines (1 line = 12 points). Returns the converted measurement as a **Single**.

expression.**PointsToLines**(*Points*)

expression Optional. An expression that returns an **Application** object.

Points Required **Single**. The measurement, in points.

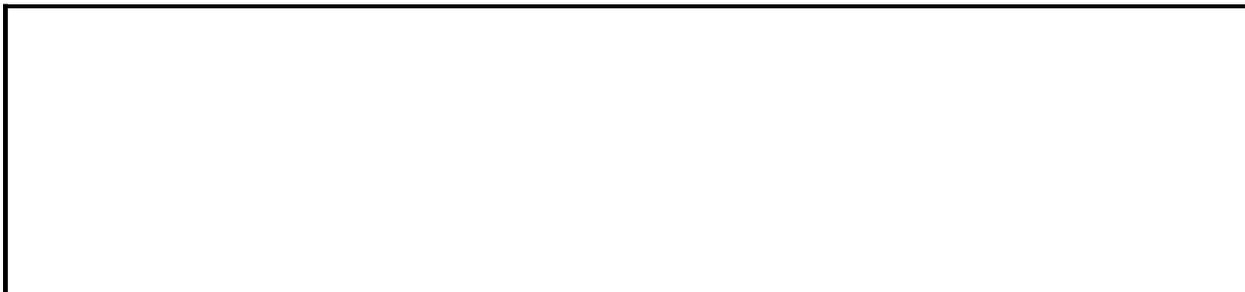
Example

This example converts the line spacing value of the first paragraph in the selection from points to lines.

```
MsgBox PointsToLines(Selection.Paragraphs(1).LineSpacing) _  
    & " lines"
```

This example converts the value of the variable `sngData` (a measurement in points) to centimeters, inches, lines, millimeters, or picas, depending on the value of the variable `intUnit` (a value from 1 through 5 that indicates the resulting unit of measurement).

```
Function ConvertPoints(ByVal intUnit As Integer, _  
    sngData As Single) As Single  
  
    Select Case intUnit  
        Case 1  
            ConvertPoints = PointsToCentimeters(sngData)  
        Case 2  
            ConvertPoints = PointsToInches(sngData)  
        Case 3  
            ConvertPoints = PointsToLines(sngData)  
        Case 4  
            ConvertPoints = PointsToMillimeters(sngData)  
        Case 5  
            ConvertPoints = PointsToPicas(sngData)  
        Case Else  
            Error 5  
    End Select  
  
End Function
```



PointsToMillimeters Method

-
Converts a measurement from points to millimeters (1 millimeter = 2.835 points). Returns the converted measurement as a **Single**.

expression.**PointsToMillimeters**(*Points*)

expression Optional. An expression that returns an **Application** object.

Points Required **Single**. The measurement, in points.

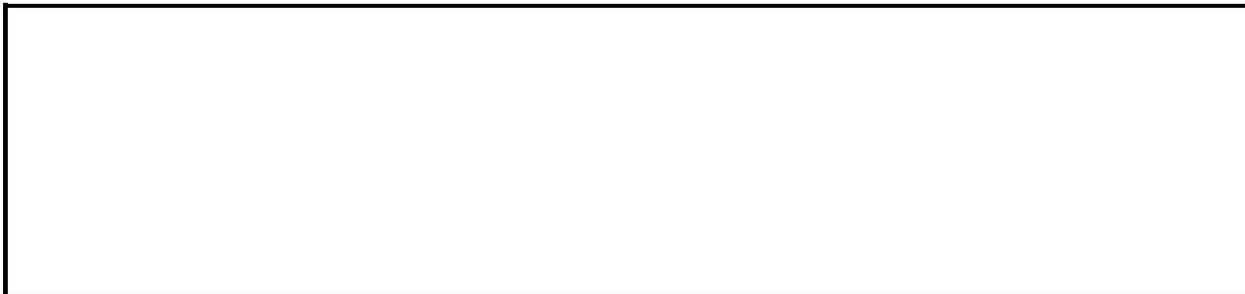
Example

This example converts 72 points to the corresponding number of millimeters.

```
MsgBox PointsToMillimeters(72) & " millimeters"
```

This example converts the value of the variable `sngData` (a measurement in points) to centimeters, inches, lines, millimeters, or picas, depending on the value of the variable `intUnit` (a value from 1 through 5 that indicates the resulting unit of measurement).

```
Function ConvertPoints(ByVal intUnit As Integer, _  
    sngData As Single) As Single  
  
    Select Case intUnit  
        Case 1  
            ConvertPoints = PointsToCentimeters(sngData)  
        Case 2  
            ConvertPoints = PointsToInches(sngData)  
        Case 3  
            ConvertPoints = PointsToLines(sngData)  
        Case 4  
            ConvertPoints = PointsToMillimeters(sngData)  
        Case 5  
            ConvertPoints = PointsToPicas(sngData)  
        Case Else  
            Error 5  
    End Select  
  
End Function
```



PointsToPicas Method

-
Converts a measurement from points to picas (1 pica = 12 points). Returns the converted measurement as a **Single**.

expression.**PointsToPicas**(*Points*)

expression Optional. An expression that returns an **Application** object.

Points Required **Single**. The measurement, in points.

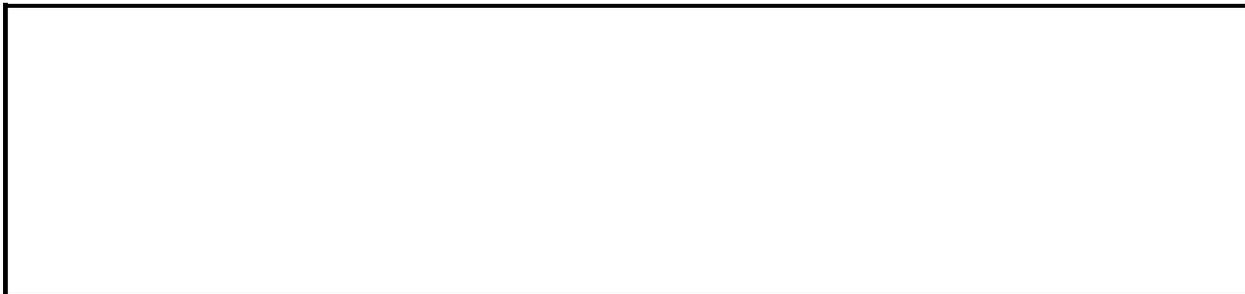
Example

This example converts 36 points to the corresponding number of picas.

```
MsgBox PointsToPicas(36) & " picas"
```

This example converts the value of the variable `sngData` (a measurement in points) to centimeters, inches, lines, millimeters, or picas, depending on the value of the variable `intUnit` (a value from 1 through 5 that indicates the resulting unit of measurement).

```
Function ConvertPoints(ByVal intUnit As Integer, _  
    sngData As Single) As Single  
  
    Select Case intUnit  
        Case 1  
            ConvertPoints = PointsToCentimeters(sngData)  
        Case 2  
            ConvertPoints = PointsToInches(sngData)  
        Case 3  
            ConvertPoints = PointsToLines(sngData)  
        Case 4  
            ConvertPoints = PointsToMillimeters(sngData)  
        Case 5  
            ConvertPoints = PointsToPicas(sngData)  
        Case Else  
            Error 5  
    End Select  
  
End Function
```



PointsToPixels Method

-

Converts a measurement from points to pixels. Returns the converted measurement as a **Single**.

expression.**PointsToPixels**(*Points*, *fVertical*)

expression Required. An expression that returns an **Application** object.

Points Required **Single**. The point value to be converted to pixels.

fVertical Optional **Variant**. **True** to return the result as vertical pixels; **False** to return the result as horizontal pixels.

Example

This example displays the height and width in pixels of an object measured in points.

```
MsgBox "180x120 points is equivalent to " _  
    & PointsToPixels(180, False) & "x" _  
    & PointsToPixels(120, True) _  
    & " pixels on this display."
```



Post Method

-

Posts the specified document to a public folder in Microsoft Exchange. This method displays the **Send to Exchange Folder** dialog box so that a folder can be selected.

expression.**Post**

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

Example

This example displays the **Send to Exchange Folder** dialog box so that the active document can be posted to a public folder.

ActiveDocument.Post



PresentIt Method

-
Opens PowerPoint with the specified Word document loaded.

expression.**PresentIt**

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

Example

This example sends a copy of the document named "MyPresentation.doc" to PowerPoint.

```
Documents("MyPresentation.doc").PresentIt
```



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 a **CalloutFormat** object.

DropType Required [MsoCalloutDropType](#). The starting position of the callout line relative to the text bounding box. If you specify **msoCalloutDropCustom**, the values of the [Drop](#) and [AutoAttach](#) properties and the relative positions of the callout text box and callout line origin (the place that the callout points to) are used to determine where the callout line attaches to the text box.

MsoCalloutDropType can be one of these MsoCalloutDropType constants.

msoCalloutDropCenter

msoCalloutDropMixed

msoCalloutDropBottom

msoCalloutDropCustom

msoCalloutDropTop

Example

This example specifies that the callout line attach to the top of the text bounding box for the first shape on the active document. For the example to work, the first shape must be a callout.

```
ActiveDocument.Shapes(1).Callout.PresetDrop msoCalloutDropTop
```

This example toggles between two preset drops for the first shape on the active document. For the example to work, the first shape must be a callout.

```
With ActiveDocument.Shapes(1).Callout
  If .DropType = msoCalloutDropTop Then
    .PresetDrop msoCalloutDropBottom
  ElseIf .DropType = msoCalloutDropBottom Then
    .PresetDrop msoCalloutDropTop
  End If
End With
```



PresetGradient Method

Sets the specified fill to a preset gradient.

expression.**PresetGradient**(*Style*, *Variant*, *PresetGradientType*)

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

Style Required [MsoGradientStyle](#). The gradient style.

MsoGradientStyle can be one of these MsoGradientStyle constants.

msoGradientDiagonalDown

msoGradientDiagonalUp

msoGradientFromCenter

msoGradientFromCorner

msoGradientFromTitle Only used in Microsoft PowerPoint.

msoGradientHorizontal

msoGradientMixed

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 **msoGradientFromCenter**, this argument can be either 1 or 2.

PresetGradientType Required [MsoPresetGradientType](#). The gradient type.

MsoPresetGradientType can be one of these MsoPresetGradientType constants.

msoGradientBrass

msoGradientChrome

msoGradientDaybreak

msoGradientEarlySunset

msoGradientFog

msoGradientGoldII
msoGradientLateSunset
msoGradientMoss
msoGradientOcean
msoGradientPeacock
msoGradientRainbowII
msoGradientSilver
msoGradientWheat
msoPresetGradientMixed
msoGradientCalmWater
msoGradientChromeII
msoGradientDesert
msoGradientFire
msoGradientGold
msoGradientHorizon
msoGradientMahogany
msoGradientNightfall
msoGradientParchment
msoGradientRainbow
msoGradientSapphire

Example

This example adds a rectangle with a preset gradient fill to the active document.

```
ActiveDocument.Shapes.AddShape( _  
    msoShapeRectangle, 90, 90, 140, 80).Fill.PresetGradient _  
    msoGradientHorizontal, 1, msoGradientBrass
```



PresetTextured Method

Sets the specified fill to a preset texture.

expression.**PresetTextured**(*PresetTexture*)

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

PresetTexture Required [MsoPresetTexture](#). The preset texture.

MsoPresetTexture can be one of these MsoPresetTexture constants.

msoPresetTextureMixed

msoTextureBouquet

msoTextureCanvas

msoTextureDenim

msoTextureGranite

msoTextureMediumWood

msoTextureOak

msoTexturePapyrus

msoTexturePinkTissuePaper

msoTextureRecycledPaper

msoTextureStationery

msoTextureWaterDroplets

msoTextureWovenMat

msoTextureBlueTissuePaper

msoTextureBrownMarble

msoTextureCork

msoTextureFishFossil

msoTextureGreenMarble

msoTextureNewsprint

msoTexturePaperBag

msoTextureParchment
msoTexturePurpleMesh
msoTextureSand
msoTextureWalnut
msoTextureWhiteMarble

Example

This example adds a rectangle with a green-marble textured fill to the active document.

```
ActiveDocument.Shapes.AddShape(msoShapeCan, 90, 90, 40, 80) _  
    .Fill.PresetTextured msoTextureGreenMarble
```



↳ [Show All](#)

Previous Method

▶ [Previous method as it applies to the **Paragraph** object.](#)

Returns the previous paragraph as a **Paragraph** object.

expression.**Previous**(*Count*)

expression Required. An expression that returns one of the above objects.

Count Optional **Variant**. The number of paragraphs by which you want to move back. The default value is 1.

▶ [Previous method as it applies to the **Range** and **Selection** objects.](#)

Returns a **Range** object relative to the specified selection or range.

Note If the **Range** or **Selection** is just after the specified **Unit** the **Range** or **Selection** is moved to the previous unit. For example, if the **Selection** is just after a word (before the trailing space), the following instruction moves the **Selection** backwards to the previous word.

```
Selection.Previous(Unit:=wdWord, Count:=1).Select
```

expression.**Previous**(*Unit*, *Count*)

expression Required. An expression that returns one of the above objects.

Unit Optional **Variant**. [WdUnits](#)

Can be one of the following **WdUnits** constants.

wdCharacter

wdWord

wdSentence

wdParagraph

wdSection

wdStory

wdCell

wdColumn

wdRow

wdTable

If *expression* returns a **Selection** object, **wdLine** can also be used. The default value is **wdCharacter**.

Count Optional **Variant**. The number of units by which you want to move back. The default value is 1.

► [Previous method as it applies to the **Browser** object.](#)

For the **Browser** object, moves the selection to the previous item indicated by the browser target. Use the [Target](#) property to change the browser target.

expression.**Previous**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Browser** object.](#)

This example moves the insertion point into the first cell (the cell in the upper-left corner) of the previous table.

```
With Application.Browser  
    .Target = wdBrowseTable  
    .Previous  
End With
```

▶ [As it applies to the **Paragraph** object.](#)

This example selects the paragraph that precedes the selection in the active document.

```
Selection.Previous(Unit:=wdParagraph, Count:=1).Select
```

▶ [As it applies to the **Range** object.](#)

This example applies bold formatting to the first word in the active document.

```
ActiveDocument.Words(2) _  
    .Previous(Unit:=wdWord, Count:=1).Bold = True
```



PreviousField Method

-
Selects the previous field. If a field is found, this method returns a **Field** object; if not, it returns **Nothing**.

expression.**PreviousField**

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

Example

This example updates the previous field (the field immediately preceding the selection).

```
If Not (Selection.PreviousField Is Nothing) Then  
    Selection.Fields.Update  
End If
```

This example selects the previous field, and if a field is found, displays a message in the status bar.

```
Set myField = Selection.PreviousField  
If Not (myField Is Nothing) Then StatusBar = "Field found"
```



PreviousHeaderFooter Method

-

If the selection is in a header, this method moves to the previous header within the current section (for example, from an even header to an odd header) or to the last header in the previous section. If the selection is in a footer, this method moves to the previous footer.

Note If the selection is in the first header or footer in the first section of the document, or if it's not in a header or footer at all, an error occurs.

expression.**PreviousHeaderFooter**

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

Example

This example inserts an even section break, switches the active window to print layout view, displays the current header, and then switches to the previous header.

```
Selection.Collapse Direction:=wdCollapseStart
Selection.InsertBreak Type:=wdSectionBreakEvenPage
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .SeekView = wdSeekCurrentPageHeader
    .PreviousHeaderFooter
End With
```



PreviousRevision Method

-
Locates and returns the previous tracked change as a **Revision** object.

expression.**PreviousRevision**(*Wrap*)

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

Wrap Optional **Variant**. **True** to continue searching for a revision at the end of the document when the beginning of the document is reached. The default value is **False**.

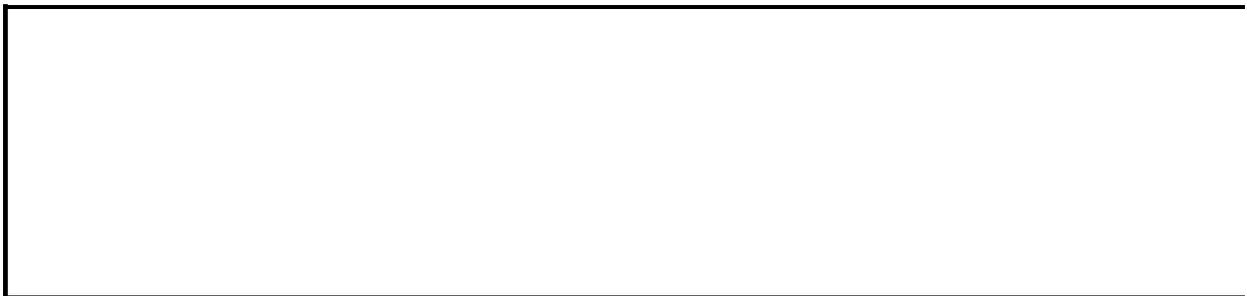
Example

This example selects the last tracked change in the first section in the active document and displays the date and time of the change.

```
Selection.EndOf Unit:=wdStory, Extend:=wdMove
Set myRev = Selection.PreviousRevision
If Not (myRev Is Nothing) Then MsgBox myRev.Date
```

This example rejects the previous tracked change found if the change type is deleted or inserted text. If the tracked change is a style change, the change is accepted.

```
Set myRev = Selection.PreviousRevision(Wrap:=True)
If Not (myRev Is Nothing) Then
    Select Case myRev.Type
        Case wdRevisionDelete
            myRev.Reject
        Case wdRevisionInsert
            myRev.Reject
        Case wdRevisionStyle
            myRev.Accept
    End Select
End If
```



PreviousSubdocument Method

-
Moves the range or selection to the previous subdocument. If there isn't another subdocument, an error occurs.

expression.**PreviousSubdocument**

expression Required. An expression that returns a **Range** or **Selection** object.

Example

This example switches the active document to master document view and selects the previous subdocument.

```
If ActiveDocument.Subdocuments.Count >= 1 Then
    ActiveDocument.ActiveWindow.View.Type = wdMasterView
    Selection.EndKey Unit:=wdStory, Extend:=wdMove
    Selection.PreviousSubdocument
End If
```



PrevNode Method

Returns a [DiagramNode](#) object that represents the previous diagram node in a collection of diagram nodes.

expression.**PrevNode**

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

Remarks

Use the [NextNode](#) method to return the next **DiagramNode** object in a collection of diagram nodes.

Example

This example adds child nodes to the first child node in a newly-created diagram.

```
Sub AddToPrevNode()  
    Dim dgnRoot As DiagramNode  
    Dim shpDiagram As Shape  
    Dim dgnPrev As DiagramNode  
    Dim intCount As Integer  
  
    'Add organizational chart to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, _  
         Left:=10, _  
         Top:=15, _  
         Width:=400, _  
         Height:=475)  
  
    'Add first diagram node  
    Set dgnRoot = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes off the first diagram node  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next intCount  
  
    'Access the node immediately preceding the second  
    'diagram node and add three child nodes  
    Set dgnPrev = dgnRoot.Children.Item(2).PrevNode  
  
    For intCount = 1 To 3  
        dgnPrev.Children.AddNode  
    Next intCount  
  
End Sub
```



↳ [Show All](#)

PrintOut Method

► [PrintOut method as it applies to the Application, Document, and Window objects.](#)

Prints all or part of the specified document.

expression.**PrintOut**(*Background*, *Append*, *Range*, *OutputFileName*, *From*, *To*, *Item*, *Copies*, *Pages*, *PageType*, *PrintToFile*, *Collate*, *FileName*, *ActivePrinterMacGX*, *ManualDuplexPrint*, *PrintZoomColumn*, *PrintZoomRow*, *PrintZoomPaperWidth*, *PrintZoomPaperHeight*)

expression Required. An expression that returns one of the above objects.

Background Optional **Variant**. Set to **True** to have the macro continue while Microsoft Word prints the document.

Append Optional **Variant**. Set to **True** to append the specified document to the file name specified by the **OutputFileName** argument. **False** to overwrite the contents of **OutputFileName**.

Range Optional **Variant**. The page range. Can be any [WdPrintOutRange](#) constant.

wdPrintAllDocument

wdPrintCurrentPage

wdPrintFromTo

wdPrintRangeOfPages

wdPrintSelection

OutputFileName Optional **Variant**. If **PrintToFile** is **True**, this argument specifies the path and file name of the output file.

From Optional **Variant**. The starting page number when **Range** is set to **wdPrintFromTo**.

To Optional **Variant**. The ending page number when **Range** is set to **wdPrintFromTo**.

Item Optional **Variant**. The item to be printed. Can be any [WdPrintOutItem](#) constant.

wdPrintAutoTextEntries

wdPrintComments

wdPrintDocumentContent

wdPrintKeyAssignments

wdPrintProperties

wdPrintStyles

Copies Optional **Variant**. The number of copies to be printed.

Pages Optional **Variant**. The page numbers and page ranges to be printed, separated by commas. For example, "2, 6-10" prints page 2 and pages 6 through 10.

PageType Optional **Variant**. The type of pages to be printed. Can be any [WdPrintOutPages](#) constant.

wdPrintAllPages

wdPrintEvenPagesOnly

wdPrintOddPagesOnly

PrintToFile Optional **Variant**. **True** to send printer instructions to a file. Make sure to specify a file name with **OutputFileName**.

Collate Optional **Variant**. When printing multiple copies of a document, **True** to print all pages of the document before printing the next copy.

FileName Optional **Variant**. The path and file name of the document to be printed. If this argument is omitted, Word prints the active document. (Available only with the **Application** object.)

ActivePrinterMacGX Optional **Variant**. This argument is available only in Microsoft Office Macintosh Edition. For additional information about this

argument, consult the language reference Help included with Microsoft Office Macintosh Edition.

ManualDuplexPrint Optional **Variant**. **True** to print a two-sided document on a printer without a duplex printing kit. If this argument is **True**, the [PrintBackground](#) and [PrintReverse](#) properties are ignored. Use the [PrintOddPagesInAscendingOrder](#) and [PrintEvenPagesInAscendingOrder](#) properties to control the output during manual duplex printing. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

PrintZoomColumn Optional **Variant**. The number of pages you want Word to fit horizontally on one page. Can be 1, 2, 3, or 4. Use with the **PrintZoomRow** argument to print multiple pages on a single sheet.

PrintZoomRow Optional **Variant**. The number of pages you want Word to fit vertically on one page. Can be 1, 2, or 4. Use with the **PrintZoomColumn** argument to print multiple pages on a single sheet.

PrintZoomPaperWidth Optional **Variant**. The width to which you want Word to scale printed pages, in twips (20 twips = 1 point; 72 points = 1 inch).

PrintZoomPaperHeight Optional **Variant**. The height to which you want Word to scale printed pages, in twips (20 twips = 1 point; 72 points = 1 inch).

► [PrintOut method as it applies to the Envelope object.](#)

Prints an envelope without adding the envelope to the active document.

expression.**PrintOut**(*ExtractAddress*, *Address*, *AutoText*, *OmitReturnAddress*, *ReturnAddress*, *ReturnAutoText*, *PrintBarCode*, *PrintFIMA*, *Size*, *Height*, *Width*, *FeedSource*, *AddressFromLeft*, *AddressFromTop*, *ReturnAddressFromLeft*, *ReturnAddressFromTop*, *DefaultFaceUp*, *DefaultOrientation*, *PrintEPostage*, *Vertical*, *RecipientNamefromLeft*, *RecipientNamefromTop*, *RecipientPostalfromLeft*, *RecipientPostalfromTop*, *SenderNamefromLeft*, *SenderNamefromTop*, *SenderPostalfromLeft*, *SenderPostalfromTop*)

expression Required. An expression that returns an **Envelope** object.

ExtractAddress Optional **VARIANT**. **True** to use the text marked by the "EnvelopeAddress" bookmark (a user-defined bookmark) as the recipient's address.

Address Optional **VARIANT**. A string that specifies the recipient's address (ignored if **ExtractAddress** is **True**).

AutoText Optional **VARIANT**. The name of the AutoText entry that includes a recipient's address.

OmitReturnAddress Optional **VARIANT**. **True** to omit the return address.

ReturnAddress Optional **VARIANT**. A string that specifies the return address.

ReturnAutoText Optional **VARIANT**. The name of the AutoText entry that includes a return address.

PrintBarcode Optional **VARIANT**. **True** to add a POSTNET bar code. For U.S. mail only.

PrintFIMA Optional **VARIANT**. **True** to add a Facing Identification Mark (FIM-A) for use in presorting courtesy reply mail. For U.S. mail only.

Size Optional **VARIANT**. A string that specifies the envelope size. The string should match one of the sizes listed on the left side of the Envelope size box in the **Envelope Options** dialog box (for example, "Size 10").

Height Optional **VARIANT**. The height of the envelope (in points) when the **Size** argument is set to "Custom size."

Width Optional **VARIANT**. The width of the envelope (in points) when the **Size** argument is set to "Custom size."

FeedSource Optional **VARIANT**. **True** to use the [FeedSource](#) property of the [Envelope](#) object to specify which paper tray to use when printing the envelope.

AddressFromLeft Optional **VARIANT**. The distance (in points) between the left edge of the envelope and the recipient's address.

AddressFromTop Optional **VARIANT**. The distance (in points) between the top

edge of the envelope and the recipient's address.

ReturnAddressFromLeft Optional **Variant**. The distance (in points) between the left edge of the envelope and the return address.

ReturnAddressFromTop Optional **Variant**. The distance (in points) between the top edge of the envelope and the return address.

DefaultFaceUp Optional **Variant**. **True** to print the envelope face up; **False** to print it face down.

DefaultOrientation Optional **Variant**. The orientation of the envelope. Can be any [WdEnvelopeOrientation](#) constant.

wdLeftPortrait

wdCenterPortrait

wdRightPortrait

wdLeftLandscape

wdCenterLandscape

wdRightLandscape

wdLeftClockwise

wdCenterClockwise

wdRightClockwise

PrintEPostage Optional **Variant**. **True** to print postage using an Internet e-postage vendor.

Vertical Optional **Variant**. **True** prints text vertically on the envelope. Used for Asian-language envelopes.

RecipientNamefromLeft Optional **Variant**. The position of the recipient's name, measured in points, from the left edge of the envelope. Used for Asian-language envelopes.

RecipientNamefromTop Optional **Variant**. The position of the recipient's name, measured in points, from the top edge of the envelope. Used for Asian-language envelopes.

RecipientPostalfromLeft Optional **VARIANT**. The position of the recipient's postal code, measured in points, from the left edge of the envelope. Used for Asian-language envelopes.

RecipientPostalfromTop Optional **VARIANT**. The position of the recipient's postal code, measured in points, from the top edge of the envelope. Used for Asian-language envelopes.

SenderNamefromLeft Optional **VARIANT**. The position of the sender's name, measured in points, from the left edge of the envelope. Used for Asian-language envelopes.

SenderNamefromTop Optional **VARIANT**. The position of the sender's name, measured in points, from the top edge of the envelope. Used for Asian-language envelopes.

SenderPostalfromLeft Optional **VARIANT**. The position of the sender's postal code, measured in points, from the left edge of the envelope. Used for Asian-language envelopes.

SenderPostalfromTop Optional **VARIANT**. The position of the sender's postal code, measured in points, from the top edge of the envelope. Used for Asian-language envelopes.

► [PrintOut method as it applies to the **MailingLabel** object.](#)

Prints a label or a page of labels with the same address.

expression.**PrintOut**(*Name*, *Address*, *ExtractAddress*, *LaserTray*, *SingleLabel*, *Row*, *Column*, *PrintEPostageLabel*, *Vertical*)

expression Required. An expression that returns a [MailingLabel](#) object.

Name Optional **VARIANT**. The mailing label name.

Address Optional **VARIANT**. The text for the label address.

ExtractAddress Optional **VARIANT**. **True** to use the text marked by the "EnvelopeAddress" bookmark (a user-defined bookmark) as the label text. If this argument is specified, ***Address*** and ***AutoText*** are ignored.

LaserTray Optional **VARIANT**. The laser printer tray to be used. Can be any [WdPaperTray](#) constant.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormatBin

wdPrinterTractorFeed

wdPrinterUpperBin

SingleLabel Optional **VARIANT**. **True** to print a single label; **False** to print an entire page of the same label.

Row Optional **VARIANT**. The label row for a single label. Not valid if **SingleLabel** is **False**.

Column Optional **VARIANT**. The label column for a single label. Not valid if **SingleLabel** is **False**.

PrintEPostageLabel Optional **VARIANT**. **True** to print postage using an Internet e-postage vendor.

Vertical Optional **VARIANT**. **True** prints text vertically on the label. Used for Asian-language mailing labels.

Example

▶ [As it applies to the **Application**, **Document**, and **Window** objects.](#)

This example prints the current page of the active document.

```
ActiveDocument.PrintOut Range:=wdPrintCurrentPage
```

This example prints all the documents in the current folder. The **Dir** function is used to return all file names that have the file name extension ".doc".

```
adoc = Dir("*.DOC")
Do While adoc <> ""
    Application.PrintOut FileName:=adoc
    adoc = Dir()
Loop
```

This example prints the first three pages of the document in the active window.

```
ActiveDocument.ActiveWindow.PrintOut _
    Range:=wdPrintFromTo, From:="1", To:="3"
```

This example prints the comments in the active document.

```
If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.PrintOut Item:=wdPrintComments
End If
```

This example prints the active document, fitting six pages on each sheet.

```
ActiveDocument.PrintOut PrintZoomColumn:=3, _
    PrintZoomRow:=2
```

This example prints the active document at 75% of actual size.

```
ActiveDocument.PrintOut _
    PrintZoomPaperWidth:=0.75 * (8.5 * 1440), _
    PrintZoomPaperHeight:=0.75 * (11 * 1440)
```

▶ [As it applies to the **Envelope** object.](#)

This example prints an envelope using the user address as the return address and

a predefined recipient address.

```
recep = "Don Funk" & vbCr & "123 Skye St." & vbCr & _  
    "OurTown, WA 98107"  
ActiveDocument.Envelope.PrintOut Address:=recep, _  
    ReturnAddress:=Application.UserAddress, _  
    Size:="Size 10", PrintBarcode:=True
```

▶ [As it applies to the **MailingLabel** object.](#)

This example prints a page of Avery 5664 mailing labels, using the specified address.

```
addr = "Jane Doe" & vbCr & "123 Skye St." _  
    & vbCr & "OurTown, WA 98107"  
Application.MailingLabel.PrintOut Name:="5664", Address:=addr
```



PrintPreview Method

-
Switches the view to print preview.

Note In addition to using the **PrintPreview** method, you can set the [PrintPreview](#) property to **True** or **False** to switch to or from print preview, respectively. You can also change the view by setting the [Type](#) property for the **View** object to **wdPrintPreview**.

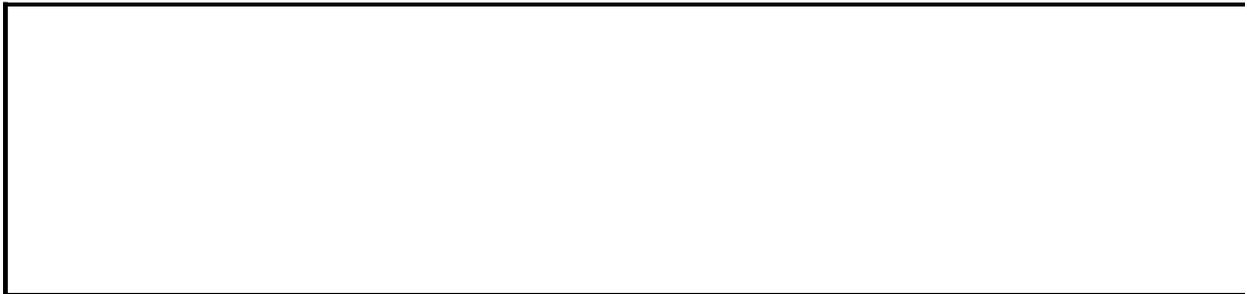
expression.**PrintPreview**

expression Required. An expression that returns an **Document** object.

Example

This example switches the active document to print preview if it's currently in some other view.

```
If Application.PrintPreview = False Then  
    ActiveDocument.PrintPreview  
End If
```



ProductCode Method

Returns the Microsoft Word globally unique identifier (GUID) as a **String**.

expression.**ProductCode**

expression Required. An expression that returns an **Application** object.

Example

This example displays the GUID for Microsoft Word.

MsgBox Application.**ProductCode**



Protect Method

Protects the specified document from changes. When a document is protected, the user can make only limited changes, such as adding annotations, making revisions, or completing a form.

Note If the document is already protected when you use this method, an error occurs.

expression.**Protect**(*Type*, *NoReset*, *Password*)

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

Type Required The protection type for the specified document.

[WdProtectionType](#).

WdProtectionType can be one of these WdProtectionType constants.

wdAllowOnlyComments

wdAllowOnlyFormFields

wdAllowOnlyRevisions

wdNoProtection

NoReset Optional **Variant**. **False** to reset form fields to their default values.

True to retain the current form field values if the specified document is protected. If **Type** isn't **wdAllowOnlyFormFields**, the **NoReset** argument is ignored.

Password Optional **Variant**. The password required to "unprotect" the specified document.

Example

This example protects the active document for forms without resetting the contents of the form fields.

```
If ActiveDocument.ProtectionType = wdNoProtection Then
    ActiveDocument.Protect _
        Type:=wdAllowOnlyFormFields, NoReset:=True
End If
```

This example protects Monthly Report.doc so that only comments can be added to it. The password "free" is required to unprotect the document.

```
Set myDoc = Documents("Monthly Report.doc")
myDoc.Protect Type:=wdAllowOnlyComments, Password:="free"
```



↳ [Show All](#)

Quit Method

Quits Microsoft Word and optionally saves or routes the open documents.

expression.Quit(**SaveChanges**, **Format**, **RouteDocument**)

expression Required. An expression that returns an [Application](#) object.

SaveChanges Optional **Variant**. Specifies whether Word saves changed documents before quitting. Can be one of the [WdSaveOptions](#) constants.

WdSaveOptions can be one of these WdSaveOptions constants.

wdDoNotSaveChanges

wdPromptToSaveChanges

wdSaveChanges

OriginalFormat Optional **Variant**. Specifies the way Word saves documents whose original format was not Word Document format. Can be one of the [WdOriginalFormat](#) constants.

WdOriginalFormat can be one of these WdOriginalFormat constants.

wdOriginalDocumentFormat

wdPromptUser

wdWordDocument

RouteDocument Optional **Variant**. **True** to route the document to the next recipient. If the document doesn't have a routing slip attached, this argument is ignored.

Example

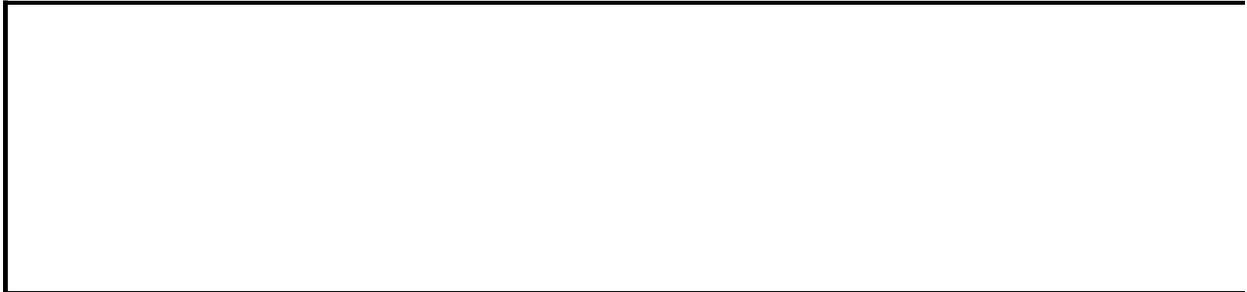
This example quits Word and prompts the user to save each document that has changed since it was last saved.

```
Application.Quit SaveChanges:=wdPromptToSaveChanges
```

This example prompts the user to save all documents. If the user clicks the Yes button, all documents are saved in the Word format before Word quits.

```
Dim intResponse As Integer
```

```
intResponse = _  
    MsgBox("Do you want to save all documents?", vbYesNo)  
If intResponse = vbYes Then Application.Quit _  
    SaveChanges:=wdSaveChanges, OriginalFormat:=wdWordDocument
```



↳ [Show All](#)

Range Method

▶ [Range method as it applies to the **Document** object.](#)

Returns a [Range](#) object by using the specified starting and ending character positions.

expression.**Range**(*Start*, *End*)

expression Required. An expression that returns a [Document](#) object.

Start Optional **Variant**. The starting character position.

End Optional **Variant**. The ending character position.

▶ [Range method as it applies to the **CanvasShapes**, **GroupShapes**, and **Shapes** objects.](#)

Returns a [ShapeRange](#) object.

expression.**Range**(*Index*)

expression Required. An expression that returns one of the above objects.

Index Required **Variant**. Specifies which shapes are to be included in the specified range. Can be an integer that specifies the index number of a shape within the [Shapes](#) collection, a string that specifies the name of a shape, or a **Variant** array that contains integers or strings.

Remarks

Character position values begin with 0 (zero) at the beginning of the document. All characters are counted, including nonprinting characters. Hidden characters are counted even if they're not displayed. If you don't specify starting and ending character positions for the **Range** method, the entire document is returned as a **Range** object.

ShapeRange objects don't include [InlineShape](#) objects. An **InlineShape** object is equivalent to a character and is positioned as a character within a range of text. [Shape](#) objects are anchored to a range of text (the selection, by default), but they can be positioned anywhere on the page. A **Shape** object will always appear on the same page as the range it's anchored to.

Most operations that you can do with a **Shape** object you can also do with a **ShapeRange** object that contains a single shape. Some operations, when performed on a **ShapeRange** object that contains multiple shapes, produce an error.

Example

▶ [As it applies to the **Document** object.](#)

This example applies bold formatting to the first 10 characters in the active document.

```
Sub DocumentRange()  
    ActiveDocument.Range(Start:=0, End:=10).Bold = True  
End Sub
```

This example creates a range that starts at the beginning of the active document and ends at the cursor position, and then it changes all characters within that range to uppercase.

```
Sub DocumentRange2()  
    Dim r As Range  
    Set r = ActiveDocument.Range(Start:=0, End:=Selection.End)  
    r.Case = wdUpperCase  
End Sub
```

This example creates and sets the variable myRange to paragraphs three through six in the active document, and then it right-aligns the paragraphs in the range.

```
Sub DocumentRange3()  
    Dim aDoc As Document  
    Dim myRange As Range  
    Set aDoc = ActiveDocument  
    If aDoc.Paragraphs.Count >= 6 Then  
        Set myRange = aDoc.Range(aDoc.Paragraphs(2).Range.Start, _  
            aDoc.Paragraphs(4).Range.End)  
        myRange.Paragraphs.Alignment = wdAlignParagraphRight  
    End If  
End Sub
```

▶ [As it applies to the **CanvasShapes**, **GroupShapes**, and **Shapes** objects.](#)

This example sets the fill foreground color to purple for the first shape in the active document.

```
Sub ShRange()  
    With ActiveDocument.Shapes.Range(1).Fill
```

```

        .ForeColor.RGB = RGB(255, 0, 255)
        .Visible = msoTrue
    End With
End Sub

```

This example applies a shadow to a variable shape in the active document.

```

Sub ShpRange2(strShpName As String)
    ActiveDocument.Shapes.Range(strShpName).Shadow.Type = msoShadow6
End Sub

```

To call the preceding subroutine, enter the following code into a standard code module.

```

Sub CallShpRange2()
    Dim shpArrow As Shape
    Dim strName As String

    Set shpArrow = ActiveDocument.Shapes.AddShape(Type:=msoShapeLeft,
        Left:=200, Top:=400, Width:=50, Height:=75)

    shpArrow.Name = "myShape"
    strName = shpArrow.Name
    ShpRange2 strShpName:=strName
End Sub

```

This example selects shapes one and three in the active document.

```

Sub SelectShapeRange()
    ActiveDocument.Shapes.Range(Array(1, 3)).Select
End Sub

```

This example selects and deletes the shapes in the first shape in the active document. This example assumes that the first shape is a canvas shape.

```

Sub CanvasShapeRange()
    Dim rngCanvasShapes As Range
    Set rngCanvasShapes = ActiveDocument.Shapes(1).CanvasItems.Range
    rngCanvasShapes.Select
    rngCanvasShapes.Delete
End Sub

```



RangeFromPoint Method

Returns the **Range** or **Shape** object that is located at the point specified by the screen position coordinate pair. If no range or shape is located at the coordinate pair specified, the method returns **Nothing**.

expression.**RangeFromPoint**(*x*, *y*)

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

x Required **Long**. The horizontal distance (in pixels) from the left edge of the screen to the point.

y Required **Long**. The vertical distance (in pixels) from the top of the screen to the point.

Example

This example creates a new document and adds a five-point star. It then obtains the screen location of the shape and calculates where the center of the shape is. Using these coordinates, the example uses the **RangeFromPoint** method to return a reference to the shape and change its fill color.

```
Dim pLeft As Long
Dim pTop As Long
Dim pWidth As Long
Dim pHeight As Long
Dim newShape As Object
Dim newDoc As New Document

With newDoc
    .Shapes.AddShape msoShape5pointStar, _
        288, 100, 100, 72
    .ActiveWindow.GetPoint pLeft, pTop, _
        pWidth, pHeight, .Shapes(1)
    Set newShape = .ActiveWindow.RangeFromPoint(pLeft _
        + pWidth * 0.5, pTop + pHeight * 0.5)
    newShape.Fill.ForeColor.RGB = RGB(80, 160, 130)
End With
```



Rebind Method

Changes the command assigned to the specified key binding.

expression.**Rebind**(**KeyCategory**, **Command**, **CommandParameter**)

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

KeyCategory Required [WdKeyCategory](#). The key category of the specified key binding.

WdKeyCategory can be one of these WdKeyCategory constants.

wdKeyCategoryAutoText

wdKeyCategoryCommand

wdKeyCategoryDisable

wdKeyCategoryFont

wdKeyCategoryMacro

wdKeyCategoryNil

wdKeyCategoryPrefix

wdKeyCategoryStyle

wdKeyCategorySymbol

Command Required **String**. The name of the specified command.

CommandParameter Optional **Variant**. Additional text, if any, required for the command specified by **Command**. For information about values for this argument, see the [Add](#) method for the **KeyBindings** object.

Example

This example reassigns the CTRL+SHIFT+S key binding to the **FileSaveAs** command.

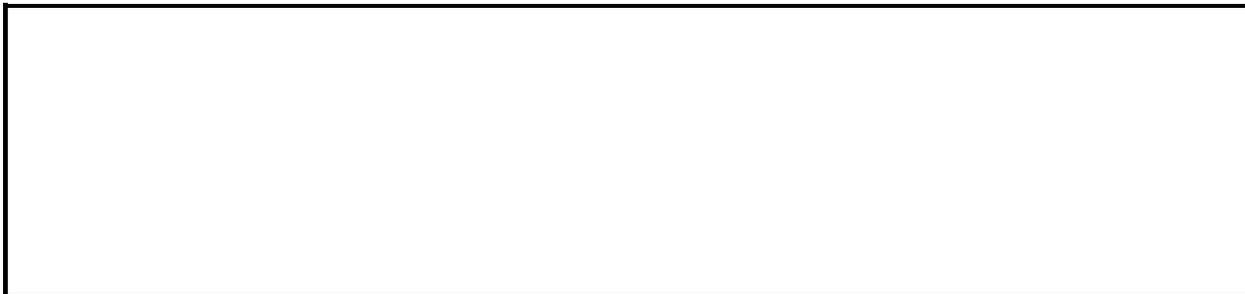
```
Dim kbTemp As KeyBinding

CustomizationContext = NormalTemplate
Set kbTemp = _
    FindKey(BuildKeyCode(wdKeyControl, wdKeyShift, wdKeyS))
kbTemp.Rebind KeyCategory:=wdKeyCategoryCommand, _
    Command:="FileSaveAs"
```

This example rebinds all keys assigned to the macro named "Macro1" to the macro named "ReportMacro."

```
Dim kbLoop As KeyBinding

CustomizationContext = ActiveDocument.AttachedTemplate
For Each kbLoop In _
    KeysBoundTo(KeyCategory:=wdKeyCategoryMacro, _
        Command:="Macro1")
    kbLoop.Rebind KeyCategory:=wdKeyCategoryMacro, _
        Command:="ReportMacro"
Next kbLoop
```



RecheckSmartTags Method

-
Removes smart tags recognized by the grammar checker and rechecks the document content against all smart tag recognizers.

expression.**RecheckSmartTags**

expression Required. An expression that returns a [Document](#) object.

Example

This example removes the existing smart tags in the active document and rechecks the document content against the smart tag recognizers selected on the **Smart Tags** tab of the **AutoCorrect** dialog box.

```
Sub SmartTagRecheck()  
    ActiveDocument.RecheckSmartTags  
End Sub
```



Redo Method

-
Redoes the last action that was undone (reverses the [Undo](#) method). Returns **True** if the actions were redone successfully.

expression.**Redo**(*Times*)

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

Times Optional **Variant**. The number of actions to be redone.

Example

This example redoes the last two actions in the Sales.doc redo list.

```
Documents("Sales.doc").Redo 2
```

This example redoes the last action in the active document. If the action is successfully redone, a message is displayed in the status bar.

```
On Error Resume Next  
If ActiveDocument.Redo = False Then _  
    StatusBar = "Redo was unsuccessful"
```



Reject Method

-

Rejects the specified tracked change. The revision marks are removed, leaving the original text intact.

Note Formatting changes cannot be rejected.

expression.**Reject**

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

Example

This example rejects the next tracked change found in the active document.

```
Dim revNext As Revision

If ActiveDocument.Revisions.Count >= 1 Then
    Set revNext = Selection.NextRevision
    If Not (revNext Is Nothing) Then revNext.Reject
End If
```

This example rejects the tracked changes in the first paragraph.

```
Dim rngTemp As Range
Dim revLoop As Revision

Set rngTemp = ActiveDocument.Paragraphs(1).Range
For Each revLoop In rngTemp.Revisions
    revLoop.Reject
Next revLoop
```

This example rejects the first tracked change in the selection.

```
Dim rngTemp As Range

Set rngTemp = Selection.Range
If rngTemp.Revisions.Count >= 1 Then _
    rngTemp.Revisions(1).Reject
```



RejectAll Method

-

Rejects all the tracked changes in a range. The revision marks are removed, leaving the original text intact.

expression.**RejectAll**

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

Remarks

Use the [RejectAllRevisions](#) method to reject all revisions in a document. Formatting changes cannot be rejected.

Example

This example rejects all the tracked changes in the active document.

```
ActiveDocument.Revisions.RejectAll
```

This example rejects all the tracked changes in the selection.

```
Dim rngTemp As Range
```

```
Set rngTemp = Selection.Range
```

```
rngTemp.Revisions.RejectAll
```



RejectAllRevisions Method

-
Rejects all tracked changes in the specified document.

expression.**RejectAllRevisions**

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

Example

This example checks the main story in active document for tracked changes, and if there are any, the example rejects all revisions in all stories in the document.

```
If ActiveDocument.Revisions.Count >= 1 Then _  
    ActiveDocument.RejectAllRevisions
```



RejectAllRevisionsShown Method

-
Rejects all revisions in a document that are displayed on the screen.

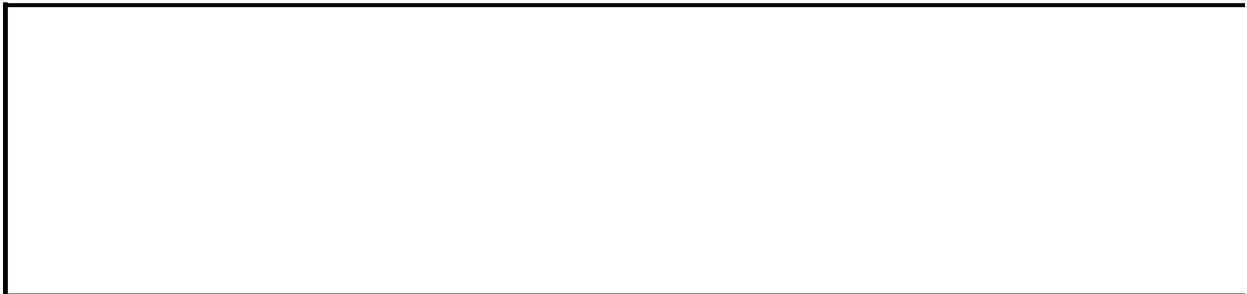
expression.**RejectAllRevisionsShown**

expression Required. An expression that returns a [Document](#) object.

Example

This example hides revisions made by Jeff Smith and rejects all remaining revisions that are displayed.

```
Sub RejectAllChanges()  
    Dim rev As Reviewer  
    With ActiveWindow.View  
        'Show all revisions in the document  
        .ShowRevisionsAndComments = True  
        .ShowFormatChanges = True  
        .ShowInsertionsAndDeletions = True  
  
        For Each rev In .Reviewers  
            rev.Visible = True  
        Next  
  
        'Hide revisions made by "Jeff Smith"  
        .Reviewers(Index:="Jeff Smith").Visible = False  
    End With  
  
    'Reject all revisions displayed in the active view  
    ActiveDocument.RejectAllRevisionsShown  
End Sub
```



Reload Method

-

Reloads a cached document by resolving the hyperlink to the document and downloading it.

Note This method reloads the document asynchronously; that is, statements following the **Reload** method in your procedure may execute before the document is actually reloaded. Because of this, you may get unexpected results from using this method in your macros.

expression.**Reload**

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

Example

This example opens and reloads the hyperlink to the address "main" on a local intranet.

```
With ActiveDocument  
    .FollowHyperlink Address:="http://main"  
    .Reload  
End With
```



ReloadAs Method

Reloads a document based on an HTML document, using the specified document encoding.

expression.**ReloadAs**(*Encoding*)

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

Encoding Required [MsoEncoding](#).

MsoEncoding can be one of these MsoEncoding constants.

msoEncodingOEMMultilingualLatinI

msoEncodingOEMNordic

msoEncodingOEMTurkish

msoEncodingSimplifiedChineseAutoDetect

msoEncodingT61

msoEncodingTaiwanEten

msoEncodingTaiwanTCA

msoEncodingTaiwanWang

msoEncodingTraditionalChineseAutoDetect

msoEncodingTurkish

msoEncodingUnicodeLittleEndian

msoEncodingUTF7

msoEncodingVietnamese

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDICKoreanExtendedAndKorean

msoEncodingEBCDICMultilingualROECELatin2

msoEncodingEBCDICSerbianBulgarian

msoEncodingEBCDICThai
msoEncodingEBCDICTurkishLatin5
msoEncodingEBCDICUSCanada
msoEncodingEBCDICUSCanadaAndTraditionalChinese
msoEncodingOEMModernGreek
msoEncodingOEMMultilingualLatinII
msoEncodingOEMPortuguese
msoEncodingOEMUnitedStates
msoEncodingSimplifiedChineseGBK
msoEncodingTaiwanCNS
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTeleText
msoEncodingThai
msoEncodingTraditionalChineseBig5
msoEncodingUnicodeBigEndian
msoEncodingUSASCII
msoEncodingUTF8
msoEncodingWestern
msoEncodingArabic
msoEncodingArabicASMO
msoEncodingArabicAutoDetect
msoEncodingArabicTransparentASMO
msoEncodingAutoDetect
msoEncodingBaltic
msoEncodingCentralEuropean
msoEncodingCyrillic
msoEncodingCyrillicAutoDetect
msoEncodingEBCDICArabic
msoEncodingEBCDICDenmarkNorway
msoEncodingEBCDICFinlandSweden
msoEncodingEBCDICFrance
msoEncodingEBCDICGermany
msoEncodingEBCDICGreek

msoEncodingEBCDICGreekModern
msoEncodingEBCDICHebrew
msoEncodingEBCDICIcelandic
msoEncodingEBCDICInternational
msoEncodingEBCDICItaly
msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese
msoEncodingEBCDIKKoreanExtended
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICRussian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICTurkish
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
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

Example

This example reloads the current document with Cyrillic encoding.

```
ActiveDocument.ReloadAs msoEncodingCyrillic
```



Relocate Method

-

In outline view, moves the paragraphs within the specified range after the next visible paragraph or before the previous visible paragraph. Body text moves with a heading only if the body text is collapsed in outline view or if it's part of the range.

expression.**Relocate**(*Direction*)

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

Direction Required [WdRelocate](#). The direction of the move.

Can be either of the following **WdRelocate** constants.

wdRelocateUp

wdRelocateDown

Example

This example moves the third, fourth, and fifth paragraphs in the active document below the next (sixth) paragraph.

```
theStart = ActiveDocument.Paragraphs(3).Range.Start
theEnd = ActiveDocument.Paragraphs(5).Range.End
Set myRange = ActiveDocument.Range(Start:=theStart, End:=theEnd)
ActiveDocument.ActiveWindow.View.Type = wdOutlineView
myRange.Relocate Direction:=wdRelocateDown
```

This example moves the first paragraph in the selection above the previous paragraph.

```
ActiveDocument.ActiveWindow.View.Type = wdOutlineView
Selection.Paragraphs(1).Range.Relocate Direction:=wdRelocateUp
```



↳ [Show All](#)

RemoveNumbers Method

-
Removes numbers or bullets from the specified **Document**, **List**, or **ListFormat** object.

expression.**RemoveNumbers**(*NumberType*)

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

NumberType Optional [WdNumberType](#). The type of number to be removed.

Can be one of the following **WdNumberType** constants.

wdNumberParagraph

wdNumberListNum

wdNumberAllNumbers

The default value is **wdNumberAllNumbers**

Remarks

When this method is applied to a **List** object, it removes numbers only from paragraphs in the specified list, skipping over any interleaved numbers from other lists. If this method is applied to the **ListFormat** object for a range of text, all numbers from all lists in the range are removed.

Example

▶ [As it applies to the **ListFormat** object.](#)

This example removes the bullets or numbers from any numbered paragraphs in the selection.

```
Selection.Range.ListFormat.RemoveNumbers
```

This example removes the LISTNUM fields from the selection.

```
Selection.Range.ListFormat.RemoveNumbers wdNumberListNum
```

▶ [As it applies to the **Document** object.](#)

This example removes the numbers from the beginning of any numbered paragraphs in the active document.

```
ActiveDocument.RemoveNumbers wdNumberParagraph
```

This example removes the bullets or numbers from the third list in MyDocument.doc.

```
If Documents("MyDocument.doc").Lists.Count >= 3 Then  
    Documents("MyDocument.doc").Lists(3).RemoveNumbers  
End If
```



RemoveSmartTags Method

-
Removes all smart tag information from a document.

expression.**RemoveSmartTags**

expression Required. An expression that returns a [Document](#) object.

Example

This example removes all smart tag information from the active document.

```
Sub SmartTagRemove()  
    ActiveDocument.RemoveSmartTags  
End Sub
```



↳ [Show All](#)

RemoveTheme Method

-
Removes the active [theme](#) from the current document.

expression.**RemoveTheme**

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

Example

This example removes the active theme from the current document.

ActiveDocument.**RemoveTheme**



Repaginate Method

-
Repaginates the entire document.

expression.**Repaginate**

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

Example

This example repaginates the active document if it's changed since the last time it was saved.

```
If ActiveDocument.Saved = False Then ActiveDocument.Repaginate
```

This example repaginates all open documents.

```
For Each doc In Documents  
    doc.Repaginate  
Next doc
```



Repeat Method

-

Repeats the most recent editing action one or more times. Returns **True** if the commands were repeated successfully.

Note Using this method is the equivalent to using the **Repeat** command on the **Edit** menu.

expression.**Repeat**(*Times*)

expression Optional. An expression that returns an **Application** object.

Times Optional **Variant**. The number of times you want to repeat the last command.

Example

This example inserts the text "Hello" followed by two paragraphs (the second typing action is repeated once).

```
Selection.TypeText "Hello"  
Selection.TypeParagraph  
Repeat
```

This example repeats the last command three times (if it can be repeated).

```
On Error Resume Next  
If Repeat(3) = True Then StatusBar = "Action repeated"
```



ReplaceNode Method

-

Replaces a target diagram node with the source diagram node. The target diagram node is deleted, and the source diagram node, including any of its child nodes, are moved to where the target diagram node was.

expression.**ReplaceNode**(*TargetNode*)

expression Required. An expression that returns a [DiagramNode](#) object.

TargetNode Required **DiagramNode** object. The diagram node to be replaced.

Example

The following example replaces the fourth diagram node of a newly-created diagram with the second node.

```
Sub Replace()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add first child node to diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three more child nodes  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    'Replace fourth node with the second node  
    dgnNode.Diagram.Nodes(2).ReplaceNode _  
        TargetNode:=dgnNode.Diagram.Nodes(4)  
  
End Sub
```



Reply Method

-

Opens a new e-mail message — with the sender's address on the **To:** line — for replying to the active message.

expression.**Reply**

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

Example

This example opens a new e-mail message for replying to the active message.

`Application.MailMessage.Reply`



ReplyAll Method

-

Opens a new e-mail message — with the sender's and all other recipients' addresses on the **To:** and **Cc:** lines, as appropriate — for replying to the active message.

expression.**ReplyAll**

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

Example

This example opens a new e-mail message for replying to the active message.

```
Application.MailMessage.ReplyAll
```



ReplyWithChanges Method

-

Sends an e-mail message to the author of a document that has been sent out for review, notifying them that a reviewer has completed review of the document.

expression.**ReplyWithChanges**(*ShowMessage*)

expression Required. An expression that returns a [Document](#) object.

ShowMessage Optional **Variant**. **True** to display the message prior to sending. **False** to automatically send the message without displaying it first. The default value is **True**.

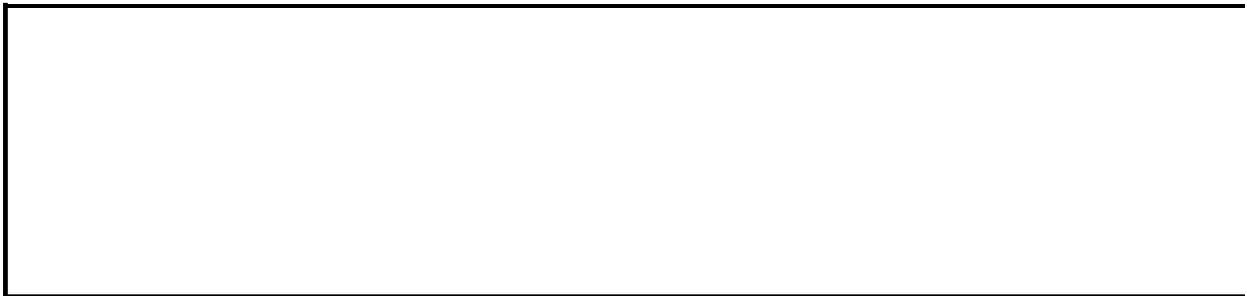
Remarks

Use the [SendForReview](#) method to start a collaborative review of a document. If the **ReplyWithChanges** method is executed on a document that is not part of a collaborative review cycle, Microsoft Word displays an error message.

Example

This example sends a message notifying the author that a reviewer has completed a review, without first displaying the e-mail message to the reviewer. This example assumes that the current document is part of a collaborative review cycle.

```
Sub ReplyMsg()  
    ThisDocument.ReplyWithChanges ShowMessage:=False  
End Sub
```



↳ [Show All](#)

Reset Method

▶ [Reset method as it applies to the **ListGallery** object.](#)

Resets the list template specified by **Index** for the specified list gallery to the built-in list template format.

expression.**Reset**(**Index**)

expression Required. An expression that returns one of the above objects.

Index Required **Long**.

▶ [Reset method as it applies to the **Font**, **InlineShape**, **Paragraph**, **ParagraphFormat**, **Paragraphs**, and **RoutingSlip** objects.](#)

Font object: Removes manual character formatting (formatting not applied using a style). For example, if you manually format a word as bold and the underlying style is plain text (not bold), the **Reset** method removes the bold format.

Paragraph, **Paragraphs**, or **ParagraphFormat** object: Removes manual paragraph formatting (formatting not applied using a style). For example, if you manually right align a paragraph and the underlying style has a different alignment, the **Reset** method changes the alignment to match the formatting of the underlying style.

RoutingSlip object: Resets the routing slip so that a new routing can be initiated with the same recipient list and delivery information. The routing must be completed before you use this method.

InlineShape object: Removes changes that were made to an inline shape.

expression.**Reset**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Font** object.](#)

This example removes manual formatting from the selection.

```
Selection.Font.Reset
```

▶ [As it applies to the **Paragraph** object.](#)

This example removes manual paragraph formatting from the second paragraph in the active document.

```
ActiveDocument.Paragraphs(2).Reset
```

▶ [As it applies to the **RoutingSlip** object.](#)

This example prepares the active document to be rerouted to the same recipients as in the previous routing settings.

```
If ActiveDocument.HasRoutingSlip = True Then  
    ActiveDocument.RoutingSlip.Reset  
End If
```

▶ [As it applies to the **InlineShape** object.](#)

This example inserts a picture as an inline shape, changes the brightness, and then resets the picture to its original brightness.

```
Set aInLine = ActiveDocument.InlineShapes.AddPicture _  
    (FileName:="C:\Windows\Bubbles.bmp", Range:=Selection.Range)  
aInLine.PictureFormat.Brightness = 0.5  
MsgBox "Changing brightness back"  
aInLine.Reset
```

▶ [As it applies to the **ListGalleries** object.](#)

This example sets the fourth format listed on the **Numbered** tab in the **Bullets and Numbering** dialog box back to the built-in numbering format, and then it applies the list template to the selection.

```
ListGalleries(wdNumberGallery).Reset(4)
Selection.Range.ListFormat.ApplyListTemplate _
    ListTemplate:=ListGalleries(2).ListTemplates(4)
```

This example resets all the list templates in the **Bullets and Numbering** dialog box back to the built-in formats.

```
For Each lg In ListGalleries
    For i = 1 to 7
        lg.Reset Index:=i
    Next i
Next lg
```



ResetContinuationNotice Method

-

Resets the footnote or endnote continuation notice to the default notice. The default notice is blank (no text).

expression.**ResetContinuationNotice**

expression Required. An expression that returns an **Endnotes** or **Footnotes** object.

Example

This example resets the endnote continuation notice for the active document.

```
ActiveDocument.Endnotes.ResetContinuationNotice
```

This example resets the footnote continuation notice and sets the starting number for footnote reference marks to 2 in Sales.doc.

```
With Documents("Sales.doc").Sections(1).Range.Footnotes  
    .ResetContinuationNotice  
    .NumberingRule = wdRestartContinuous  
    .StartingNumber = 2  
End With
```



ResetContinuationSeparator Method

-

Resets the footnote or endnote continuation separator to the default separator. The default separator is a long horizontal line that separates document text from notes continued from the previous page.

expression.**ResetContinuationSeparator**

expression Required. An expression that returns an **Endnotes** or **Footnotes** object.

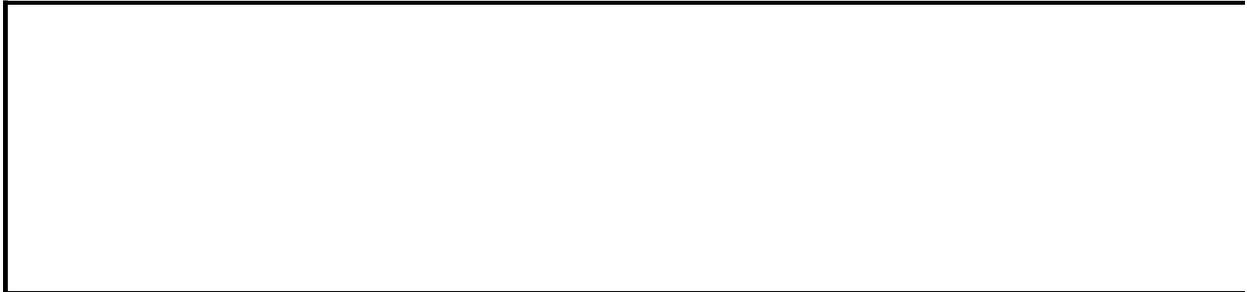
Example

This example resets the footnote continuation separator to the default separator line.

```
ActiveDocument.Footnotes.ResetContinuationSeparator
```

This example resets the endnote continuation separator for the first section in each open document.

```
Dim docLoop As Document  
  
For Each docLoop In Documents  
    docLoop.Sections(1).Range.Endnotes _  
        .ResetContinuationSeparator  
Next docLoop
```



ResetFormFields Method

-
Clears all form fields in a document, preparing the form to be filled in again.

expression.**ResetFormFields**

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

Remarks

Use the **ResetFormFields** method to clear fields when a document's fields are not locked. To clear fields when a document's fields are locked, use the [Protect](#) method.

Example

This example clears the fields in the active document. This example assumes that the active document contains form fields.

```
Sub ClearFormFields()  
    ActiveDocument.ResetFormFields  
End Sub
```



ResetIgnoreAll Method

-

Clears the list of words that were previously ignored during a spelling check. After you run this method, previously ignored words are checked along with all the other words.

expression.**ResetIgnoreAll**

expression Required. An expression that returns an **Application** object.

Remarks

In order for the **ResetIgnoreAll** method to work, you must first set the **SpellingChecked** property to **False**.

Example

This example clears the list of words that were ignored during a previous spelling check and then begins a new spelling check on the active document.

```
Application.ResetIgnoreAll  
ActiveDocument.SpellingChecked = False  
ActiveDocument.CheckSpelling
```



ResetRotation Method

-

Resets the extrusion rotation around the x-axis and the y-axis to 0 (zero) so that the front of the extrusion faces forward. This method doesn't reset the rotation around the z-axis.

expression.**ResetRotation**

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

Remarks

To set the extrusion rotation around the x-axis and the y-axis to anything other than 0 (zero), 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 0 (zero) for the extrusion of the first shape on the active document.

```
ActiveDocument.Shapes(1).ThreeD.ResetRotation
```



ResetSeparator Method

-

Resets the footnote or endnote separator to the default separator. The default separator is a short horizontal line that separates document text from notes.

expression.**ResetSeparator**

expression Required. An expression that returns an **Endnotes** or **Footnotes** object.

Example

This example resets the footnote separator to the default separator line.

```
ActiveDocument.Footnotes.ResetSeparator
```

This example resets the endnote separator for the notes in the document where the selection is located.

```
Selection.Endnotes.ResetSeparator
```



Resize Method

-

Sizes the Word application window or the specified task window. If the window is maximized or minimized, an error occurs.

Note Use the [Width](#) or [Height](#) property to set the window width and height independently.

expression.Resize(Width, Height)

expression Required. An expression that returns an **Application** or **Task** object.

Width Required **Long**. The width of the window, in points.

Height Required **Long**. The height of the window, in points.

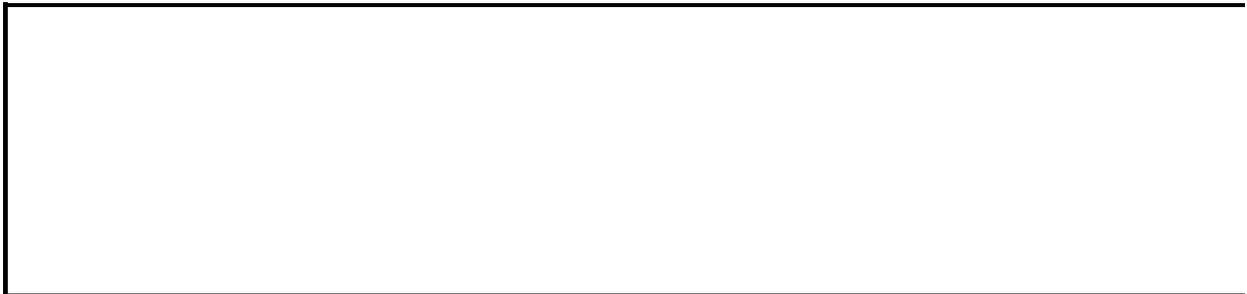
Example

This example resizes the Microsoft Excel application window to 6 inches wide by 4 inches high.

```
If Tasks.Exists("Microsoft Excel") = True Then
    With Tasks("Microsoft Excel")
        .WindowState = wdWindowStateNormal
        .Resize Width:=InchesToPoints(6), Height:=InchesToPoints(4)
    End With
End If
```

This example resizes the Word application window to 7 inches wide by 6 inches high.

```
With Application
    .WindowState = wdWindowStateNormal
    .Resize Width:=InchesToPoints(7), Height:=InchesToPoints(6)
End With
```



Route Method

-

Routes the specified document, using the document's current routing slip.

Remarks

If the document doesn't have a routing slip, an error occurs. Use the [HasRoutingSlip](#) property to determine whether there's a routing slip attached to the document. Routing a document sets the [Routed](#) property to **True**.

expression.**Route**

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

Example

If the active document has a routing slip attached to it, this example routes the document.

```
If ActiveDocument.HasRoutingSlip = True Then ActiveDocument.Route
```

This example routes Feedback.doc to two recipients, one after the other.

```
Documents("Feedback.doc").HasRoutingSlip = True  
With Documents("Feedback.doc").RoutingSlip  
    .Subject = "Your feedback please..."  
    .AddRecipient Recipient:="Tad Orman"  
    .AddRecipient Recipient:="David Simpson"  
    .Delivery = wdOneAfterAnother  
End With  
Documents("Status.doc").Route
```



RtlPara Method

-
Sets the reading order and alignment of the specified paragraphs to right-to-left.

expression.**RtlPara**

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

Remarks

For all selected paragraphs, this method sets the reading order to right-to-left. If a paragraph with a left-to-right reading order is also left-aligned, this method reverses its reading order and sets its paragraph alignment to right-aligned.

Use the [ReadingOrder](#) property to change the reading order without affecting paragraph alignment.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the reading order and alignment of the current selection to right-to-left if the selection isn't styled as "Normal."

```
If Selection.Style <> "Normal" Then _  
    Selection.RtlPara
```



↳ [Show All](#)

RtlRun Method

-
Sets the reading order and alignment of the specified [run](#) to right-to-left.

expression.**RtlRun**

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

Remarks

For the specified run, this method sets the reading order to right-to-left. If a paragraph in the run with a left-to-right reading order is also left-aligned, this method reverses its reading order and sets its paragraph alignment to right-aligned.

Use the [ReadingOrder](#) property to change the reading order without affecting paragraph alignment.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the reading order and alignment of the specified run to right-to-left if the selection isn't styled as "Normal."

```
If Selection.Style <> "Normal" Then _  
    Selection.Rt1Run
```



Run Method

Runs a Visual Basic macro.

expression.**Run**(*MacroName*, *varg1*, *varg2*, *varg3*, *varg4*, *varg5*, *varg6*, *varg7*, *varg8*, *varg9*, *varg10*, *varg11*, *varg12*, *varg13*, *varg14*, *varg15*, *varg16*, *varg17*, *varg18*, *varg19*, *varg20*, *varg21*, *varg22*, *varg23*, *varg24*, *varg25*, *varg26*, *varg27*, *varg28*, *varg29*, *varg30*)

expression Required. An expression that returns an **Application** object.

MacroName Required **String**. The name of the macro. Can be any combination of template, module, and macro name. For example, the following statements are all valid.

```
Application.Run "Normal.Module1.MAIN"  
Application.Run "MyProject.MyModule.MyProcedure"  
Application.Run "'My Document.doc'!ThisModule.ThisProcedure"
```

If you specify the document name, your code can only run macros in documents related to the current context — not just any macro in any document.

varg1...varg30 Optional **Variant**. Macro parameter values. You can pass up to 30 parameter values to the specified macro.

Remarks

Although Visual Basic code can call a macro directly (without this method being used), this method is useful when the macro name is stored in a variable (for more information, see the example for this topic). The following statements are functionally equivalent.

```
Normal.Module2.Macro1
```

```
Call Normal.Module2.Macro1
```

```
Application.Run MacroName:="Normal.Module2.Macro1"
```

Example

This example prompts the user to enter a template name, module name, macro name, and parameter value, and then it runs that macro.

```
Dim strTemplate As String
Dim strModule As String
Dim strMacro As String
Dim strParameter As String

strTemplate = InputBox("Enter the template name")
strModule = InputBox("Enter the module name")
strMacro = InputBox("Enter the macro name")
strParameter = InputBox("Enter a parameter value")
Application.Run MacroName:=strTemplate & "." _
    & strModule & "." & strMacro, _
    varg1:=strParameter
```



RunAutoMacro Method

-

Runs an auto macro that's stored in the specified document. If the specified auto macro doesn't exist, nothing happens.

Note Use the [Run](#) method to run any macro.

expression.**RunAutoMacro**(*Which*)

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

Which Required [WdAutoMacros](#).

WdAutoMacros can be one of these WdAutoMacros constants.

wdAutoExec

wdAutoNew

wdAutoClose

wdAutoExit

wdAutoOpen

Example

This example runs the AutoOpen macro in the active document.

```
ActiveDocument.RunAutoMacro Which:=wdAutoOpen
```



RunLetterWizard Method

Runs the Letter Wizard on the specified document.

expression.**RunLetterWizard**(*LetterContent*, *WizardMode*)

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

LetterContent Optional **Variant**. A **LetterContent** object. Any filled properties in the **LetterContent** object show up as prefilled elements in the Letter Wizard dialog boxes. If this argument is omitted, the [GetLetterContent](#) method is automatically used to get a **LetterContent** object from the specified document.

WizardMode Optional **Variant**. **True** to display the **Letter Wizard** dialog box as a series of steps with a **Next**, **Back**, and **Finish** button. **False** to display the **Letter Wizard** dialog box as if it were opened from the **Tools** menu (a properties dialog box with an **OK** button and a **Cancel** button). The default value is **True**.

Remarks

Use the [CreateLetterContent](#) method to return a [LetterContent](#) object, given various letter element properties. Use the [GetLetterContent](#) method to return a **LetterContent** object based on the contents of the specified document. You can use the resulting **LetterContent** object with the **RunLetterWizard** method to preset elements in the **Letter Wizard** dialog box.

Example

This example creates a new **LetterContent** object, sets several properties for it, and then runs the Letter Wizard by using the **RunLetterWizard** method.

```
Set myContent = New LetterContent
With myContent
    .Salutation = "Hello"
    .SalutationType = wdSalutationOther
    .SenderName = Application.UserName
    .SenderInitials = Application.UserInitials
End With
Documents.Add.RunLetterWizard _
    LetterContent:=myContent, WizardMode:=True
```

The following example uses the **CreateLetterContent** method to create a new **LetterContent** object in the active document, and then it uses this object with the **RunLetterWizard** method.

```
Set myLetter = ActiveDocument _
    .CreateLetterContent(DateFormat:="July 31, 1999", _
    IncludeHeaderFooter:=False, _
    PageDesign:="C:\MSOffice\Templates" _
    & "\Letters & Faxes\Contemporary Letter.dot", _
    LetterStyle:=wdFullBlock, Letterhead:=True, _
    LetterheadLocation:=wdLetterTop, _
    LetterheadSize:=InchesToPoints(1.5), _
    RecipientName:="Dave Edson", _
    RecipientAddress:="436 SE Main St." _
    & vbCr & "Bellevue, WA 98004", _
    Salutation:="Dear Dave,", _
    SalutationType:=wdSalutationInformal, _
    RecipientReference:="", MailingInstructions:="", _
    AttentionLine:="", Subject:="End of year report", _
    CCList:="", ReturnAddress:="", SenderName:="", _
    Closing:="Sincerely yours,", SenderCompany:="", _
    SenderJobTitle:="", SenderInitials:="", _
    EnclosureNumber:=0)
ActiveDocument.RunLetterWizard LetterContent:=myLetter
```



↳ [Show All](#)

Save Method

▶ [Save method as it applies to the **Versions** object.](#)

Saves a version of the specified document with a comment.

expression.**Save**(**Comment**)

expression Required. An expression that returns one of the above objects.

Comment Optional **Variant**.

▶ [Save method as it applies to the **Documents** object.](#)

Saves all the documents in the **Documents** collection. If a document hasn't been saved before, the **Save As** dialog box prompts the user for a file name.

expression.**Save**(**NoPrompt**, **OriginalFormat**)

expression Required. An expression that returns one of the above objects.

NoPrompt Optional **Variant**. **True** to have Word automatically save all documents. **False** to have Word prompt the user to save each document that has changed since it was last saved.

OriginalFormat Optional **Variant**. Specifies the way the documents are saved.
[WdOriginalFormat](#)

Can be one of the following **WdOriginalFormat** constants

wdOriginalDocumentFormat

wdPromptUserX

wdWordDocument

▶ [Save method as it applies to the **Document** and **Template** objects.](#)

Saves the specified document or template. If the document or template hasn't been saved before, the **Save As** dialog box prompts the user for a file name.

expression.**Save**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Document** object.](#)

This example saves the active document if it's changed since it was last saved.

```
If ActiveDocument.Saved = False Then ActiveDocument.Save
```

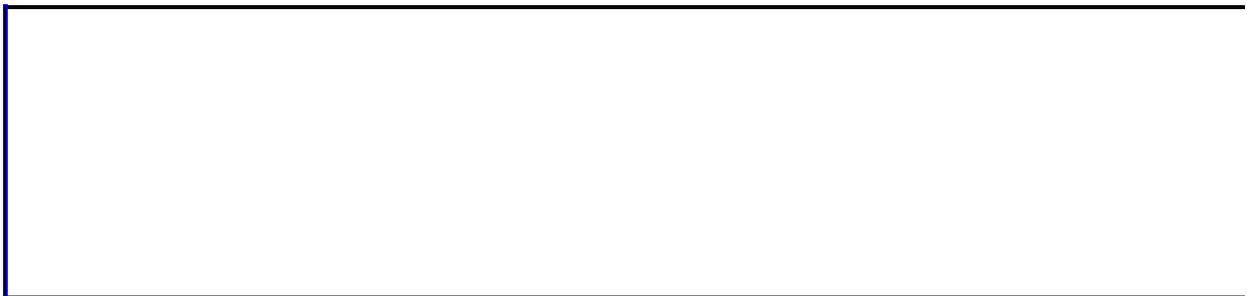
This example saves each document in the **Documents** collection without first prompting the user.

```
Documents.Save NoPrompt:=True, _  
    OriginalFormat:=wdOriginalDocumentFormat
```

▶ [As it applies to the **Version** object.](#)

If Sales.doc is open, this example saves a version of Sales.doc, with a comment.

```
For Each doc in Documents  
    If Instr(1, doc.Name, "Sales.doc", 1) > 0 Then  
        doc.Versions.Save Comment:="Minor changes to intro"  
    End If  
Next doc
```



↳ [Show All](#)

SaveAs Method

Saves the specified document with a new name or format. The arguments for this method correspond to the options in the **Save As** dialog box (**File** menu).

expression.SaveAs(FileName, FileFormat, LockComments, Password, AddToRecentFiles, WritePassword, ReadOnlyRecommended, EmbedTrueTypeFonts, SaveNativePictureFormat, SaveFormsData, SaveAsAOCELetter, Encoding, InsertLineBreaks, AllowSubstitutions, LineEnding, AddBiDiMarks)

expression Required. An expression that returns a [Document](#) object.

FileName Optional **Variant**. The name for the document. The default is the current folder and file name. If the document has never been saved, the default name is used (for example, Doc1.doc). If a document with the specified file name already exists, the document is overwritten without the user being prompted first.

FileFormat Optional **Variant**. The format in which the document is saved. Can be any [WdSaveFormat](#) constant. To save a document in another format, specify the appropriate value for the [SaveFormat](#) property of the **FileConverter** object.

WdSaveFormat can be one of these WdSaveFormat constants.

wdFormatDocument Saves as a Word document. Default.

wdFormatDOSText Saves text without formatting. Converts all section breaks, page breaks, and new line characters to paragraph marks. Uses the ANSI character set. Use this format to share documents between Word and DOS-based programs.

wdFormatDOSTextLineBreaks Saves text without formatting. Converts all line breaks, section breaks, and page breaks to paragraph marks. Use this format when you want to maintain line breaks, for example, when transferring documents to an electronic mail system.

wdFormatEncodedText Saves as an encoded text file. Use the **Encoding**

argument to specify the code page to use.

wdFormatHTML Saves all text and formatting with HTML tags so that the resulting document can be viewed in a Web browser.

wdFormatRTF Saves all formatting. Converts formatting to instructions that other programs, including compatible Microsoft programs, can read and interpret.

wdFormatTemplate Saves as a Word template.

wdFormatText Saves text without formatting. Converts all section breaks, page breaks, and new line characters to paragraph marks. Uses the ANSI character set. Use this format if the destination program cannot read any of the other available file formats.

wdFormatTextLineBreaks Saves text without formatting. Converts all line breaks, section breaks, and page breaks to paragraph marks. Use this format when you want to maintain line breaks, for example, when transferring documents to an electronic mail system.

wdFormatUnicodeText Saves as a Unicode text file. Converts text between common character encoding standards, including Unicode 2.0, Mac OS, Windows, EUC and ISO-8859 series.

Other File Types To save in a file type for which there isn't a constant, use the [FileConverters](#) object to obtain the [SaveFormat](#) property; then set the *FileFormat* argument to the value of the **SaveFormat** property.

LockComments Optional **VARIANT**. **True** to lock the document for comments. The default is **False**.

Password Optional **VARIANT**. A password string for opening the document.

AddToRecentFiles Optional **VARIANT**. **True** to add the document to the list of recently used files on the **File** menu. The default is **True**.

WritePassword Optional **VARIANT**. A password string for saving changes to the document.

ReadOnlyRecommended Optional **VARIANT**. **True** to have Microsoft Word suggest read-only status whenever the document is opened. The default is **False**.

EmbedTrueTypeFonts Optional **VARIANT**. **True** to save TrueType fonts with the document. If omitted, the *EmbedTrueTypeFonts* argument assumes the value of

the [EmbedTrueTypeFonts](#) property.

SaveNativePictureFormat Optional **Variant**. If graphics were imported from another platform (for example, Macintosh), **True** to save only the Windows version of the imported graphics.

SaveFormsData Optional **Variant**. **True** to save the data entered by a user in a form as a data record.

SaveAsAOCELetter Optional **Variant**. If the document has an attached mailer, **True** to save the document as an AOCE letter (the mailer is saved).

Encoding Optional [MsoEncoding](#). The code page, or character set, to use for documents saved as encoded text files. The default is the system code page.

MsoEncoding can be one of these MsoEncoding constants.

msoEncodingArabic

msoEncodingArabicASMO

msoEncodingArabicAutoDetect Not used with this method.

msoEncodingArabicTransparentASMO

msoEncodingAutoDetect Not used with this method.

msoEncodingBaltic

msoEncodingCentralEuropean

msoEncodingCyrillic

msoEncodingCyrillicAutoDetect Not used with this method.

msoEncodingEBCDICArabic

msoEncodingEBCDICDenmarkNorway

msoEncodingEBCDICFinlandSweden

msoEncodingEBCDICFrance

msoEncodingEBCDICGermany

msoEncodingEBCDIGreek

msoEncodingEBCDIGreekModern

msoEncodingEBCDICHebrew

msoEncodingEBCDICIcelandic

msoEncodingEBCDICInternational

msoEncodingEBCDICItaly

msoEncodingEBCDICJapaneseKatakanaExtended
msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese
msoEncodingEBCDICJapaneseLatinExtendedAndJapanese
msoEncodingEBCDIKKoreanExtended
msoEncodingEBCDIKKoreanExtendedAndKorean
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICMultilingualROECELatin2
msoEncodingEBCDICRussian
msoEncodingEBCDICSerbianBulgarian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICThai
msoEncodingEBCDICTurkish
msoEncodingEBCDICTurkishLatin5
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanada
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEBCDICUSCanadaAndTraditionalChinese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect Not used with this method.
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
msoEncodingISO2022CNSimplifiedChinese
msoEncodingISO2022CNTraditionalChinese

msoEncodingISO2022JPJISX02011989
msoEncodingISO2022JPJISX02021984
msoEncodingISO2022JPNoHalfwidthKatakana
msoEncodingISO2022KR
msoEncodingISO6937NonSpacingAccent
msoEncodingISO885915Latin9
msoEncodingISO88591Latin1
msoEncodingISO88592CentralEurope
msoEncodingISO88593Latin3
msoEncodingISO88594Baltic
msoEncodingISO88595Cyrillic
msoEncodingISO88596Arabic
msoEncodingISO88597Greek
msoEncodingISO88598Hebrew
msoEncodingISO88599Turkish
msoEncodingJapaneseAutoDetect Not used with this method.
msoEncodingJapaneseShiftJIS
msoEncodingKOI8R
msoEncodingKOI8U
msoEncodingKorean
msoEncodingKoreanAutoDetect Not used with this method.
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 Not used with this method.
msoEncodingSimplifiedChineseGBK
msoEncodingT61
msoEncodingTaiwanCNS
msoEncodingTaiwanEten
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTCA
msoEncodingTaiwanTeleText
msoEncodingTaiwanWang
msoEncodingThai
msoEncodingTraditionalChineseAutoDetect Not used with this method.
msoEncodingTraditionalChineseBig5
msoEncodingTurkish
msoEncodingUnicodeBigEndian

msoEncodingUnicodeLittleEndian
msoEncodingUSASCII
msoEncodingUTF7
msoEncodingUTF8
msoEncodingVietnamese
msoEncodingWestern

InsertLineBreaks Optional **VARIANT**. If the document is saved as a text file, **True** to insert line breaks at the end of each line of text.

AllowSubstitutions Optional **VARIANT**. If the document is saved as a text file, **True** allows Word to replace some symbols with text that looks similar. For example, displaying the copyright symbol as (c). The default is **False**.

LineEnding Optional **VARIANT**. The way Word marks the line and paragraph breaks in documents saved as text files. Can be any [WdLineEndingType](#) constant.

WdLineEndingType can be one of these WdLineEndingType constants.

wdCRLF Default.

wdCROnly

wdLFCR Not used with this method.

wdLFOnly Not used with this method.

wdLSPS Not used with this method.

AddBiDiMarks Optional **VARIANT**. **True** adds control characters to the output file to preserve bi-directional layout of the text in the original document.

Example

This example saves the active document as Test.rtf in rich-text format (RTF).

```
Sub SaveAsRTF()  
    ActiveDocument.SaveAs FileName:="Text.rtf", _  
        FileFormat:=wdFormatRTF  
End Sub
```

This example saves the active document in text-file format with the file extension ".txt".

```
Sub SaveAsTextFile()  
    Dim strDocName As String  
    Dim intPos As Integer  
  
    'Find position of extension in filename  
    strDocName = ActiveDocument.Name  
    intPos = InStrRev(strDocName, ".")  
  
    If intPos = 0 Then  
  
        'If the document has not yet been saved  
        'Ask the user to provide a filename  
        strDocName = InputBox("Please enter the name " & _  
            "of your document.")  
    Else  
  
        'Strip off extension and add ".txt" extension  
        strDocName = Left(strDocName, intPos - 1)  
        strDocName = strDocName & ".txt"  
    End If  
  
    'Save file with new extension  
    ActiveDocument.SaveAs FileName:=strDocName, _  
        FileFormat:=wdFormatText  
End Sub
```

This example loops through all the installed converters, and if it finds the WordPerfect 6.0 converter, it saves the active document using the converter.

```
Sub SaveWithConverter()  
    'Loop through all installed converters  
    'If WordPerfect 6.0 converter is found, save document using it
```

```
Dim cnvWrdPrf As FileConverter

'Look for WordPerfect file converter
'And save document using the converter
'For the FileFormat converter value
For Each cnvWrdPrf In Application.FileConverters
    If cnvWrdPrf.ClassName = "WrdPrfctWin" Then
        ActiveDocument.SaveAs FileName:="MyWP.doc", _
            FileFormat:=cnvWrdPrf.SaveFormat
    End If
Next cnvWrdPrf
```

End Sub

This example saves NewFile.doc with a write password and then closes the document. This example assumes that one of the open files is named "NewFile.doc." If not, Word displays an error message.

```
Sub SaveWithPassword()
    With Documents("NewFile.doc")
        .SaveAs WritePassword:="pass"
        .Close
    End With
End Sub
```



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. Shapes other than pictures and OLE objects are always scaled relative to their current height.

expression.ScaleHeight(*Factor*, *RelativeToOriginalSize*, *Scale*)

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**. **True** to scale the shape relative to its original size. **False** to scale it relative to its current size. You can specify **True** for this argument only if the specified shape is a picture or an OLE object.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

Scale 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

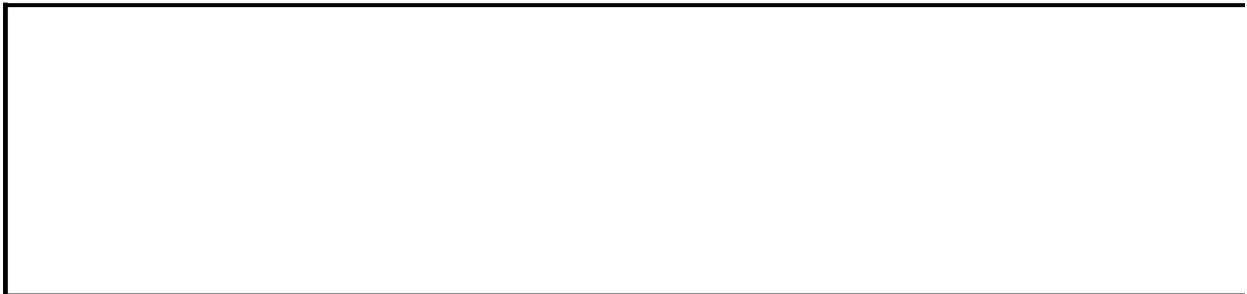
msoScaleFromTopLeft *default*

msoScaleFromMiddle

Example

This example scales all pictures and OLE objects on myDocument to 175 percent of their original height and width, and it scales all other shapes to 175 percent of their current height and width.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
  Select Case s.Type
    Case msoEmbeddedOLEObject, msoLinkedOLEObject, _
         msoOLEControlObject, _
         msoLinkedPicture, msoPicture
      s.ScaleHeight 1.75, True
      s.ScaleWidth 1.75, True
    Case Else
      s.ScaleHeight 1.75, False
      s.ScaleWidth 1.75, False
  End Select
Next
```



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. Shapes other than pictures and OLE objects are always scaled relative to their current width.

expression.ScaleWidth(**Factor**, **RelativeToOriginalSize**, **Scale**)

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**. **True** to scale the shape relative to its original size. **False** to scale it relative to its current size. You can specify **True** for this argument only if the specified shape is a picture or an OLE object.

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

Scale 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

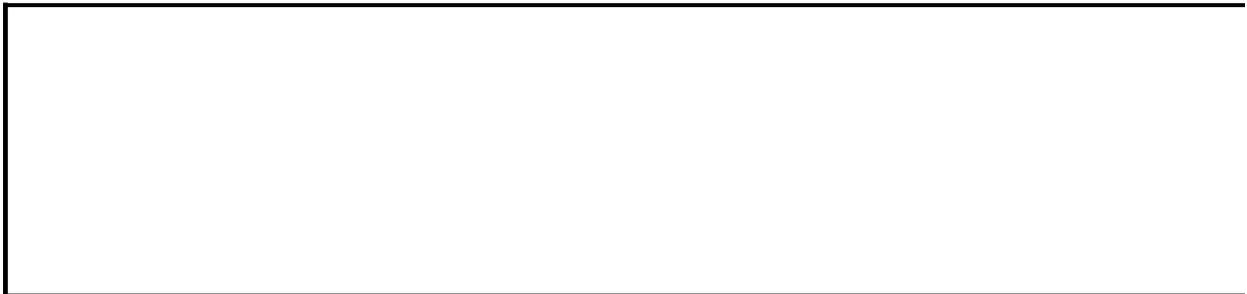
msoScaleFromTopLeft *default*

msoScaleFromMiddle

Example

This example scales all pictures and OLE objects on myDocument to 175 percent of their original height and width, and it scales all other shapes to 175 percent of their current height and width.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
  Select Case s.Type
    Case msoEmbeddedOLEObject, msoLinkedOLEObject, _
         msoOLEControlObject, _
         msoLinkedPicture, msoPicture
      s.ScaleHeight 1.75, True
      s.ScaleWidth 1.75, True
    Case Else
      s.ScaleHeight 1.75, False
      s.ScaleWidth 1.75, False
  End Select
Next
```



ScreenRefresh Method

-

Updates the display on the monitor with the current information in the video memory buffer. You can use this method after using the **ScreenUpdating** property to disable screen updates.

expression.**ScreenRefresh**

expression Required. An expression that returns an **Application** object.

Remarks

ScreenRefresh turns on screen updating for just one instruction and then immediately turns it off. Subsequent instructions don't update the screen until screen updating is turned on again with the **ScreenUpdating** property.

Example

This example turns off screen updating, opens Test.doc, inserts text, refreshes the screen, and then closes the document (with changes saved).

```
Dim rngTemp As Range
```

```
ScreenUpdating = False
```

```
Documents.Open FileName:="C:\DOCS\TEST.DOC"
```

```
Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
```

```
rngTemp.InsertBefore "new"
```

```
Application.ScreenRefresh
```

```
ActiveDocument.Close SaveChanges:=wdSaveChanges
```

```
ScreenUpdating = True
```



ScrollIntoView Method

-

Scrolls through the document window so the specified range or shape is displayed in the document window.

expression.**ScrollIntoView**(*Obj*, *Start*)

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

Obj Required **Object**. A **Range** or **Shape** object.

Start Optional **Boolean**. **True** if the top left corner of the range or shape appears at the top left corner of the document window. **False** if the bottom right corner of the range or shape appears at the bottom right corner of the document window. The default value is **True**.

Remarks

If the range or shape is larger than the document window, the ***Start*** argument specifies which portion of the range or shape displays or gets initial focus. This method cannot be used with outline view.

Example

This example scrolls through the active document so that the current selection is visible in the document window.

```
ActiveWindow.ScrollIntoView Selection.Range, True
```



↳ [Show All](#)

Select Method

- ▶ [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**. If adding a shape, **True** replaces the selection. **False** adds the new shape to the selection.

- ▶ [Select method as it applies to all other objects in the **Applies To** list.](#)

Selects the specified object.

Note After using this method, use the [Selection](#) property to work with the selected items. For more information, see [Working with the Selection object](#).

expression.**Select**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Range** object.](#)

This example selects the first paragraph in the active document.

```
Sub SelectParagraph()  
    ActiveDocument.Paragraphs(1).Range.Select  
    Selection.Font.Bold = True  
End Sub
```

▶ [As it applies to the **Row** object.](#)

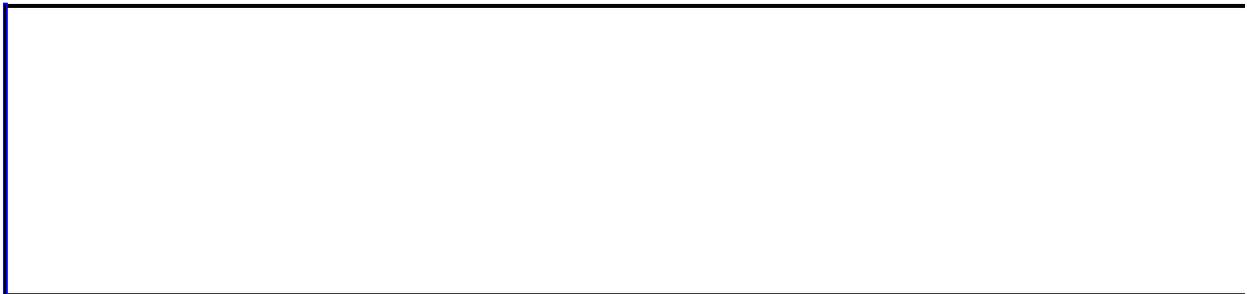
This example selects row one in table one of Report.doc.

```
Documents("Report.doc").Tables(1).Rows(1).Select
```

▶ [As it applies to the **Field** object.](#)

This example updates and selects the first field in the active document.

```
ActiveDocument.ActiveWindow.View.FieldShading = _  
    wdFieldShadingWhenSelected  
If ActiveDocument.Fields.Count >= 1 Then  
    With ActiveDocument.Fields(1)  
        .Update  
        .Select  
    End With  
End If
```



SelectAll Method

-
Selects all the shapes in the main story, in a canvas, or in headers and footers of a document.

expression.**SelectAll**

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

Remarks

This method doesn't select [InlineShape](#) objects.

Example

This example selects all the shapes in the active document.

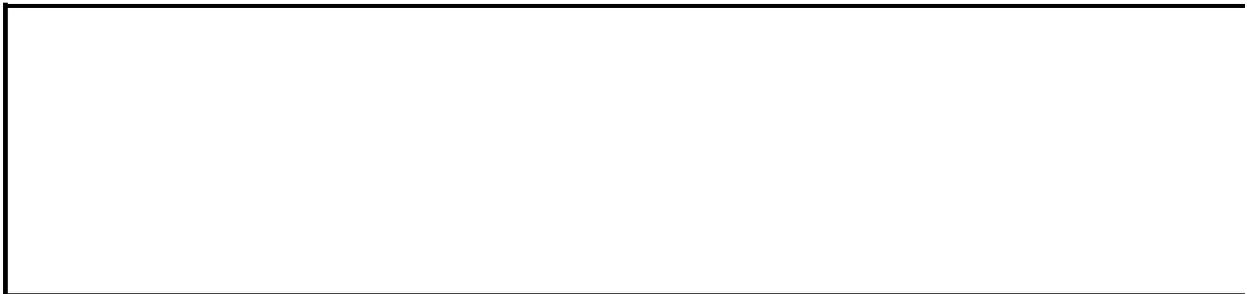
```
Sub SelectAllShapes()  
    ActiveDocument.Shapes.SelectAll  
End Sub
```

This example selects all the shapes in the headers and footers of the active document and adds a red shadow to each shape.

```
Sub SelectAllHeaderShapes()  
    With ActiveDocument.ActiveWindow  
        .View.Type = wdPrintView  
        .ActivePane.View.SeekView = wdSeekCurrentPageHeader  
    End With  
  
    ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Shapes  
  
    With Selection.ShapeRange.Shadow  
        .Type = msoShadow10  
        .ForeColor.RGB = RGB(220, 0, 0)  
    End With  
End Sub
```

This example selects and deletes all the shapes inside the first canvas of the active document.

```
Sub SelectCanvasShapes()  
    Dim s As Shape  
    Set s = ActiveDocument.Shapes.Range(1)  
    s.CanvasItems.SelectAll  
    Selection.Delete  
End Sub
```



SelectCell Method

-

Selects the entire cell containing the current selection. To use this method, the current selection must be contained within a single cell.

expression.**SelectCell**

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

Example

This example selects the entire cell containing the current selection.

Selection.**SelectCell**



SelectColumn Method

-
Selects the column that contains the insertion point, or selects all columns that contain the selection. If the selection isn't in a table, an error occurs.

expression.**SelectColumn**

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

Example

This example collapses the selection to the ending point and then selects the column that contains the insertion point.

```
Selection.Collapse Direction:=wdCollapseEnd
If Selection.Information(wdWithInTable) = True Then
    Selection.SelectColumn
End If
```



SelectCurrentAlignment Method

-
Extends the selection forward until text with a different paragraph alignment is encountered.

expression.**SelectCurrentAlignment**

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

Remarks

There are four types of paragraph alignment: left, centered, right, and justified.

Example

This example positions the insertion point at the beginning of the first paragraph after the current paragraph that doesn't have the same alignment as the current paragraph. If the alignment is the same from the selection to the end of the document, the example moves the selection to the end of the document and displays a message.

```
Selection.SelectCurrentAlignment  
Selection.Collapse Direction:=wdCollapseEnd  
If Selection.End = ActiveDocument.Content.End - 1 Then  
    MsgBox "No change in alignment found."  
End If
```



SelectCurrentColor Method

-
Extends the selection forward until text with a different color is encountered.

expression.**SelectCurrentColor**

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

Example

This example extends the selection from the beginning of the document to the first character formatted with a different color and then displays the number of characters in the resulting selection.

```
Selection.HomeKey Unit:=wdStory, Extend:=wdMove  
Selection.SelectCurrentColor  
n = Len(Selection.Text)  
MsgBox "Contiguous characters with the same color: " & n
```



SelectCurrentFont Method

-
Extends the selection forward until text in a different font or font size is encountered.

expression.**SelectCurrentFont**

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

Example

This example extends the selection until text in a different font or font size is encountered. The example uses the **Grow** method to increase the size of the selected text to the next available font size.

```
With Selection  
    .SelectCurrentFont  
    .Font.Grow  
End With
```



SelectCurrentIndent Method

-
Extends the selection forward until text with different left or right paragraph indents is encountered.

expression.**SelectCurrentIndent**

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

Example

This example jumps to the beginning of the first paragraph in the document that has different indents than the first paragraph in the active document.

```
With Selection
    .HomeKey Unit:=wdStory, Extend:=wdMove
    .SelectCurrentIndent
    .Collapse Direction:=wdCollapseEnd
End With
```

This example determines whether all the paragraphs in the active document are formatted with the same left and right indents and then displays a message box indicating the result.

```
With Selection
    .HomeKey Unit:=wdStory, Extend:=wdMove
    .SelectCurrentIndent
    .Collapse Direction:=wdCollapseEnd
End With
If Selection.End = ActiveDocument.Content.End - 1 Then
    MsgBox "All paragraphs share the same left " _
        & "and right indents."
Else
    MsgBox "Not all paragraphs share the same left " _
        & "and right indents."
End If
```



SelectCurrentSpacing Method

-
Extends the selection forward until a paragraph with different line spacing is encountered.

expression.**SelectCurrentSpacing**

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

Example

This example selects all consecutive paragraphs that have the same line spacing and changes the line spacing to single spacing.

With Selection

```
.SelectCurrentSpacing
```

```
.ParagraphFormat.Space1
```

End With



SelectCurrentTabs Method

-
Extends the selection forward until a paragraph with different tab stops is encountered.

expression.**SelectCurrentTabs**

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

Example

This example selects the second paragraph in the active document and then extends the selection to include all other paragraphs that have the same tab stops.

```
Set myRange = ActiveDocument.Paragraphs(2).Range
myRange.Select
Selection.SelectCurrentTabs
```

This example selects paragraphs that have the same tab stops and retrieves the position of the first tab stop. The example moves the selection to the next range of paragraphs that have the same tab stops. The example then adds the tab stop setting from the first group of paragraphs to the current selection.

```
With Selection
    .SelectCurrentTabs
    pos = .Paragraphs.TabStops(1).Position
    .Collapse Direction:=wdCollapseEnd
    .SelectCurrentTabs
    .Paragraphs.TabStops.Add Position:=pos
End With
```



SelectNumber Method

-
Selects the number or bullet in a list.

expression.**SelectNumber**

expression Required. An expression that returns a [Paragraph](#) object.

Remarks

If the **SelectNumber** method is called from a paragraph, selection, or range that does not contain a list, an error message is displayed.

Example

This example selects the bullet or number for the list that contains the selected paragraph in the active document, and then it increases the font size of the bullet or number to 17 points. This example assumes that the first paragraph in the selection is formatted as a bulleted or numbered list.

```
Sub SelectParaNumber()  
    With Selection  
        .Paragraphs(1).SelectNumber  
        .Font.Size = 17  
    End With  
End Sub
```



SelectRow Method

-
Selects the row that contains the insertion point, or selects all rows that contain the selection. If the selection isn't in a table, an error occurs.

expression.**SelectRow**

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

Example

This example collapses the selection to the starting point and then selects the column that contains the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart  
If Selection.Information(wdWithInTable) = True Then  
    Selection.SelectRow  
End If
```



↳ [Show All](#)

SendFax Method

▶ [SendFax method as it applies to the **Document** object.](#)

Sends the specified document as a fax, without any user interaction.

expression.**SendFax**(*Address*, *Subject*)

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

Address Required **String**. The recipient's fax number.

Subject Optional **Variant**. The text for the subject line. The character limit is 255.

▶ [SendFax method as it applies to the **Application** object.](#)

Starts the Fax Wizard.

expression.**SendFax**

expression Required. An expression that returns an **Application** object.

Example

▶ [As it applies to the **Document** object.](#)

This example sends the active document as a fax.

```
ActiveDocument.SendFax Address:="12065551234", _  
    Subject:="Important Fax"
```

▶ [As it applies to the **Application** object.](#)

This example starts the Fax Wizard.

```
Application.SendFax
```



SendForReview Method

Sends a document in an e-mail message for review by the specified recipients.

expression.**SendForReview**(*Recipients*, *Subject*, *ShowMessage*, *IncludeAttachment*)

expression Required. An expression that returns a [Document](#) object.

Recipients Optional **Variant**. A string that lists the people to whom to send the message. These can be unresolved names and aliases in an e-mail phone book or full e-mail addresses. Separate multiple recipients with a semicolon (;). If left blank and **ShowMessage** is **False**, you will receive an error message and the message will not be sent.

Subject Optional **Variant**. A string for the subject of the message. If left blank, the subject will be: Please review "*filename*".

ShowMessage Optional **Variant**. A **Boolean** value that indicates whether the message should be displayed when the method is executed. The default value is **True**. If set to **False**, the message is automatically sent to the recipients without first showing the message to the sender.

IncludeAttachment Optional **Variant**. A **Boolean** value that indicates whether the message should include an attachment or a link to a server location. The default value is **True**. If set to **False**, the document must be stored at a shared location.

Remarks

The **SendForReview** method starts a collaborative review cycle. Use the [EndReview](#) method to end a review cycle.

Example

This example automatically sends the current document as an attachment in an e-mail message to the specified recipients.

```
Sub WebReview()  
    ThisDocument.SendForReview _  
        Recipients:="someone@microsoft.com; amy jones", _  
        Subject:="Please review this document.", _  
        ShowMessage:=False, _  
        IncludeAttachment:=True  
End Sub
```



SendMail Method

-
Opens a message window for sending the specified document through Microsoft Exchange.

Note Use the [SendMailAttach](#) property to control whether the document is sent as text in the message window or as an attachment.

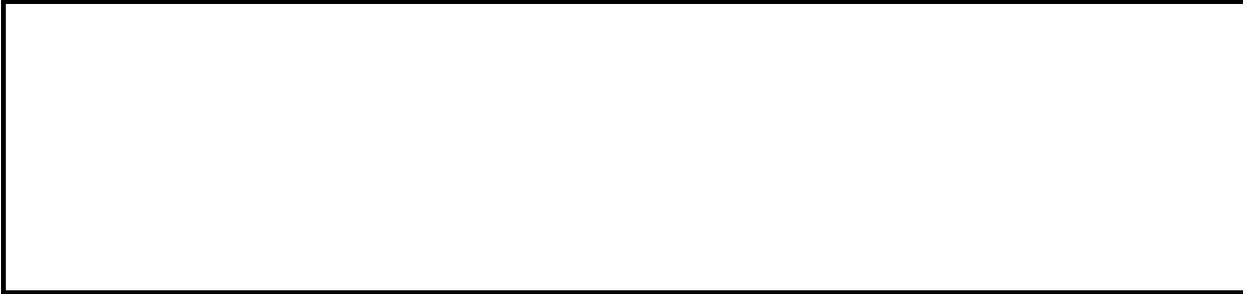
expression.**SendMail**

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

Example

This example sends the active document as an attachment to a mail message.

```
Options.SendMailAttach = True  
ActiveDocument.SendMail
```



SendWindowMessage Method

Sends a Windows message and its associated parameters to the specified task.

expression.**SendWindowMessage**(*Message*, *wParam*, *lParam*)

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

Message Required **Long**. A hexadecimal number that corresponds to the message you want to send. If you have the Microsoft Platform Software Development Kit, you can look up the name of the message in the header files (Winuser.h, for example) to find the associated hexadecimal number (precede the hexadecimal value with &h).

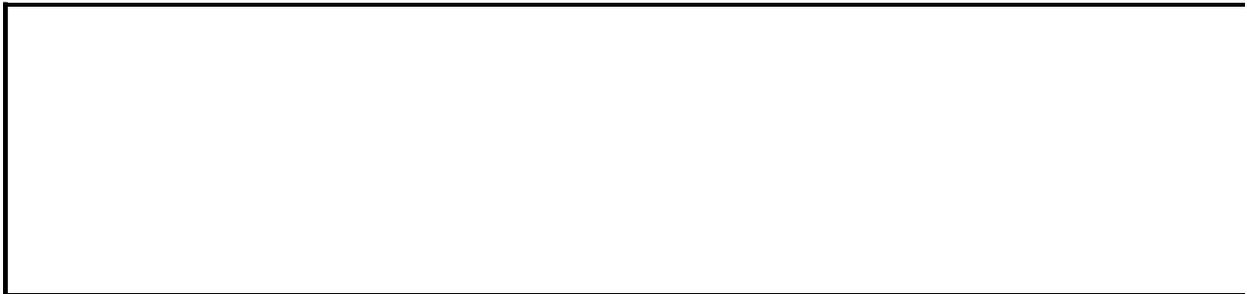
wParam, lParam Required **Long**. Parameters appropriate for the message you're sending. For information about what these values represent, see the reference topic for that message in the documentation included with the Microsoft Platform Software Development Kit. To retrieve the appropriate values, you may need to use the Spy utility (which comes with the kit).

Example

If Notepad is running, this example displays the **About** dialog box (in Notepad) by sending a WM_COMMAND message to Notepad. The **SendMessage** method is used to send the WM_COMMAND message (111 is the hexadecimal value for WM_COMMAND), with the parameters 11 and 0. The Spy utility was used to determine the *wParam* and *lParam* values.

```
Dim taskLoop As Task

For Each taskLoop In Tasks
    If InStr(taskLoop.Name, "Notepad") > 0 Then
        taskLoop.Activate
        taskLoop.SendMessage &h111, 11, 0
    End If
Next taskLoop
```



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 Required **String**. Text describing the invalid setting.

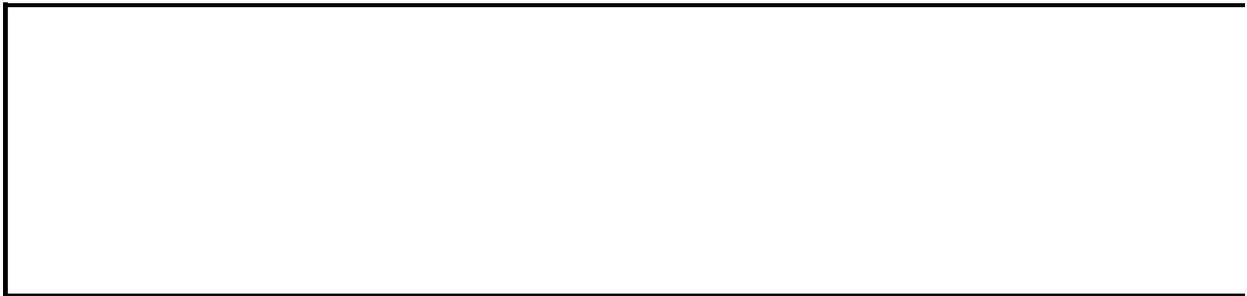
Remarks

You can individually mark data source records that contain invalid data in an address field by 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
```



SetAllFuzzyOptions Method

-
Activates all nonspecific search options associated with Japanese text.

expression.**SetAllFuzzyOptions**

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

Remarks

This method sets the following properties to **True**:

[MatchFuzzyAY](#) [MatchFuzzyBV](#)

[MatchFuzzyByte](#)

[MatchFuzzyCase](#)

[MatchFuzzyDash](#)

[MatchFuzzyDZ](#)

[MatchFuzzyHF](#)

[MatchFuzzyHiragana](#)

[MatchFuzzyIterationMark](#)

[MatchFuzzyKanji](#)

[MatchFuzzyKiKu](#)

[MatchFuzzyOldKana](#)

[MatchFuzzyProlongedSoundMark](#)

[MatchFuzzyPunctuation](#)

[MatchFuzzySmallKana](#)

[MatchFuzzySpace](#)

[MatchFuzzyTC](#)

[MatchFuzzyZJ](#)

Example

This example activates all nonspecific options before executing a search in the selected range. If the word "バイオリン" is formatted as bold, the entire paragraph is selected and copied to the Clipboard.

```
With Selection.Find
    .ClearFormatting
    .SetAllFuzzyOptions
    .Font.Bold = True
    .Execute FindText:="バイオリン", Format:=True, Forward:=True
    If .Found = True Then
        .Parent.Expand Unit:=wdParagraph
        .Parent.Copy
    End If
End With
```



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
```



SetAsTemplateDefault Method

-

Font object: Sets the specified font formatting as the default for the active document and all new documents based on the active template. The default font formatting is stored in the Normal style.

PageSetup object: Sets the specified page setup formatting as the default for the active document and all new documents based on the active template.

expression.**SetAsTemplateDefault**

expression Required. An expression that returns a **Font** or **PageSetup** object.

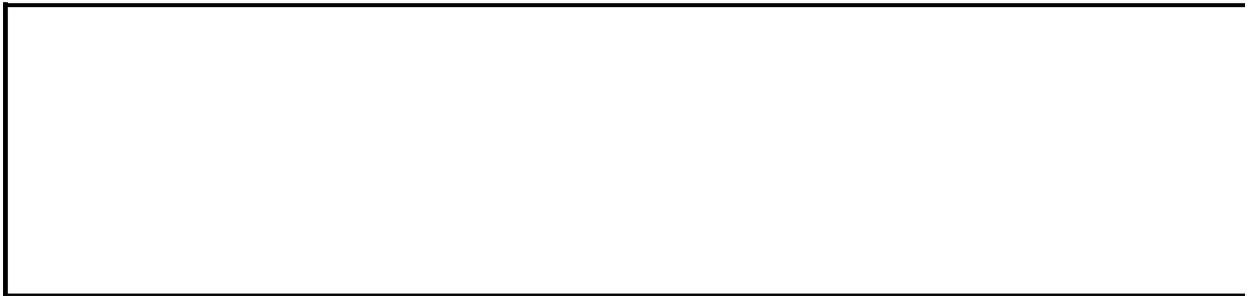
Example

This example sets the character formatting in the selection as the default.

```
Selection.Font.SetAsTemplateDefault
```

This example changes the left and right margin settings for the active document and then sets the page setup formatting as the default.

```
With ActiveDocument.PageSetup  
    .LeftMargin = InchesToPoints(1)  
    .RightMargin = InchesToPoints(1)  
    .SetAsTemplateDefault  
End With
```



SetCMYK Method

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

expression.**SetCMYK**(*Cyan, Magenta, Yellow, Black*)

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

Cyan Required **Long**. A number that represents the cyan component of the color. Can be any number from 0 to 255.

Magenta Required **Long**. A number that represents the magenta component of the color. Can be any number from 0 to 255.

Yellow Required **Long**. A number that represents the yellow component of the color. Can be any number from 0 to 255.

Black Required **Long**. A number that represents the black component of the color. Can be any number from 0 to 255.

Example

This example adds a shape to the active document and sets the CMYK fill and line colors for the specified shape.

```
Sub SetCMYKColor()  
    Dim shpHeart As Shape  
  
    Set shpHeart = ActiveDocument.Shapes.AddShape _  
        (Type:=msoShapeHeart, Left:=100, Top:=100, _  
        Width:=100, Height:=100)  
    With shpHeart  
        .Fill.ForeColor.SetCMYK Cyan:=0, _  
            Magenta:=255, Yellow:=100, Black:=0  
        .Line.ForeColor.SetCMYK Cyan:=0, _  
            Magenta:=255, Yellow:=100, Black:=20  
    End With  
End Sub
```



SetCount Method

-

Arranges text into the specified number of text columns.

Note You can also use the [Add](#) method of the **TextColumns** object to add a single column to the **TextColumns** collection.

expression.**SetCount**(*NumColumns*)

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

NumColumns Required **Long**. The number of columns the text is to be arranged into.

Example

This example arranges the text in the active document into two columns of equal width.

```
ActiveDocument.PageSetup.TextColumns.SetCount NumColumns:=2
```

This example arranges the text in the first section of Brochure.doc into three columns of equal width.

```
Documents("Brochure.doc").Sections(1) _  
    .PageSetup.TextColumns.SetCount NumColumns:=3
```



SetDefaultTableStyle Method

-
Specifies the table style to use for newly created tables in a document.

expression.**SetDefaultTableStyle**(*Style*, *SetInTemplate*)

expression Required. An expression that returns a [Document](#) object.

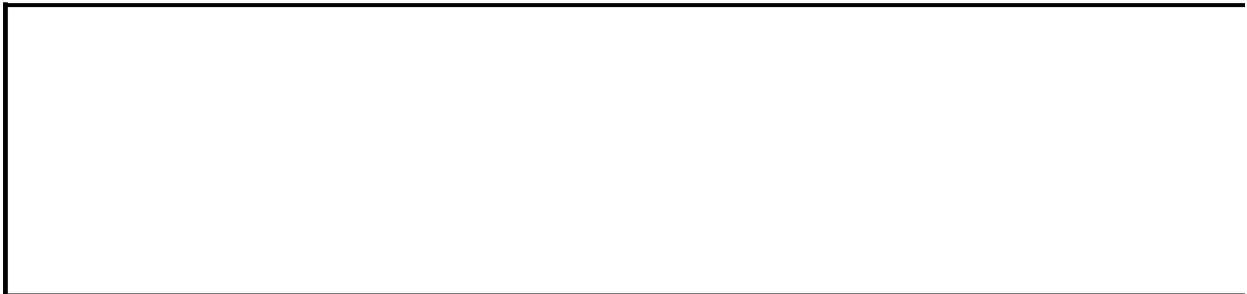
Style Required **Variant**. A string specifying the name of the style.

SetInTemplate Required **Boolean**. **True** to save the table style in the template attached to the document.

Example

This example checks to see if the default table style used in the active document is named Table Normal and, if it is, changes the default table style to TableStyle1. This example assumes that you have a table style named TableStyle1.

```
Sub TableDefaultStyle()  
    With ActiveDocument  
        If .DefaultTableStyle = "Table Normal" Then  
            .SetDefaultTableStyle _  
                Style:="TableStyle1", SetInTemplate:=True  
        End If  
    End With  
End Sub
```



↳ [Show All](#)

SetDefaultTheme Method

Sets a default [theme](#) for Microsoft Word to use with new documents, e-mail messages, or Web pages.

expression.SetDefaultTheme(*Name*, *DocumentType*)

expression Required. An expression that returns an [Application](#) object.

Name Required **String**. The name of the theme you want to assign as the default theme plus any theme formatting options you want to apply. The format of this string is "*theme nnn*" where *theme* and *nnn* are defined as follows:

String	Description
<i>theme</i>	The name of the folder that contains the data for the requested theme. (The default location for theme data folders is C:\Program Files\Common Files\Microsoft Shared\Themes.) You must use the folder name for the theme rather than the display name that appears in the Theme dialog box (Theme command, Format menu).
<i>nnn</i>	A three-digit string that indicates which theme formatting options to activate (1 to activate, 0 to deactivate). The digits correspond to the Vivid Colors , Active Graphics , and Background Image check boxes in the Theme dialog box (Theme command, Format menu). If this string is omitted, the default value for <i>nnn</i> is "011" (Active Graphics and Background Image are activated).

DocumentType Required [WdDocumentMedium](#). The type of new document to which you are assigning a default theme.

WdDocumentMedium can be one of these WdDocumentMedium constants.

wdEmailAddress

wdDocument

wdWebPage

Remarks

Setting a default theme will not apply that theme to the blank document automatically created when you start Word. Any new documents you create after that will have the default theme.

You can also use the [ThemeName](#) property to return and set the default theme for new e-mail messages.

Example

This example specifies that Microsoft Word use the Blueprint theme for all new e-mail messages.

```
Application.SetDefaultTheme "blueprnt", wdEmailMessage
```

This example specifies that Word use the Expedition theme with Active Graphics for all new Web pages.

```
Application.SetDefaultTheme "expeditn 010", wdWebPage
```



SetEditingType Method

Sets the editing type of the node specified by ***Index***. 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. Note that, depending on the editing type, this method may affect the position of adjacent nodes.

expression.**SetEditingType**(*Index*, *EditingType*)

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

Index Required **Long**. The node whose editing type is to be set.

EditingType Required [MsoEditingType](#). The editing property of the vertex.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto

msoEditingCorner

msoEditingSmooth

msoEditingSymmetric

Example

This example changes all corner nodes to smooth nodes in the third shape on the active document. The third shape must be a freeform drawing.

```
Dim lngLoop As lngLoop
```

```
With ActiveDocument.Shapes(3).Nodes
```

```
  For lngLoop = 1 to .Count
```

```
    If .Item(lngLoop).EditingType = msoEditingCorner Then
```

```
      .SetEditingType lngLoop, msoEditingSmooth
```

```
    End If
```

```
  Next lngLoop
```

```
End With
```



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 a **ThreeDFormat** object.

PresetExtrusionDirection Required [MsoPresetExtrusionDirection](#).

MsoPresetExtrusionDirection can be one of these MsoPresetExtrusionDirection constants.

msoExtrusionTop

msoExtrusionTopRight

msoExtrusionBottom

msoExtrusionBottomLeft

msoExtrusionBottomRight

msoExtrusionLeft

msoExtrusionNone

msoExtrusionRight

msoExtrusionTopLeft

msoPresetExtrusionDirectionMixed

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 on the active document extend toward the top of the shape and that the lighting for the extrusion come from the left.

```
With ActiveDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```



SetFocus Method

-

Sets the focus of the specified document window to the body of an e-mail message. If the document isn't an e-mail message, this method has no effect.

expression.**SetFocus**

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

Example

This example makes the header of an e-mail message visible and sets the focus to the body of the message.

```
ActiveWindow.EnvelopeVisible = True  
ActiveWindow.SetFocus
```



↳ [Show All](#)

SetHeight Method

▶ [SetHeight method as it applies to the Row and Rows objects.](#)

Sets the height of table rows.

expression.SetHeight(**RowHeight**, **HeightRule**)

expression Required. An expression that returns one of the above objects.

RowHeight Required **Single**. The height of the row or rows, in points.

HeightRule Required [WdRowHeightRule](#). The rule for determining the height of the specified rows.

WdRowHeightRule can be one of these WdRowHeightRule constants.

wdRowHeightAtLeast

wdRowHeightExactly

wdRowHeightAuto

▶ [SetHeight method as it applies to the Cell and Cells objects.](#)

Sets the height of table cells.

expression.SetHeight(**RowHeight**, **HeightRule**)

expression Required. An expression that returns one of the above objects.

RowHeight Required **Variant**. The height of the row or rows, in points.

HeightRule Required [WdRowHeightRule](#). The rule for determining the height of the specified cells.

WdRowHeightRule can be one of these WdRowHeightRule constants.

wdRowHeightAtLeast

wdRowHeightExactly

wdRowHeightAuto

Note: Setting the **SetHeight** property of a **Cell** or **Cells** object automatically sets the property for the entire row.

Example

▶ [As it applies to the **Rows** object.](#)

This example creates a table and then sets a fixed row height of 0.5 inch (36 points) for the first row.

```
Set newDoc = Documents.Add
Set aTable = _
    newDoc.Tables.Add(Range:=Selection.Range, NumRows:=3, _
        NumColumns:=3)
aTable.Rows(1).SetHeight RowHeight:=InchesToPoints(0.5), _
    HeightRule:=wdRowHeightExactly
```

▶ [As it applies to the **Cells** object.](#)

This example sets the row height of the selected cells to at least 18 points.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells.SetHeight RowHeight:=18, _
        HeightRule:=wdRowHeightAtLeast
Else
    MsgBox "The insertion point is not in a table."
End If
```



SetLeftIndent Method

Sets the indentation for a row or rows in a table.

expression.**SetLeftIndent**(*LeftIndent*, *RulerStyle*)

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

LeftIndent Required **Single**. The distance (in points) between the current left edge of the specified row or rows and the desired left edge.

RulerStyle Required [WdRulerStyle](#). Controls the way Word adjusts the table when the left indent is changed.

WdRulerStyle can be one of these WdRulerStyle constants.

wdAdjustNone Adjusts the left edge of row or rows, preserving the width of all columns by shifting them to the left or right. This is the default value.

wdAdjustSameWidth Adjusts the left edge of the first column, preserving the position of the right edge of the table by setting the widths of all the cells in the specified row or rows to the same value.

wdAdjustFirstColumn Adjusts the left edge of the first column only, preserving the positions of the other columns and the right edge of the table.

wdAdjustProportional Adjusts the left edge of the first column, preserving the position of the right edge of the table by proportionally adjusting the widths of all the cells in the specified row or rows.

Remarks

The **WdRulerStyle** behavior described above applies to left-aligned tables. The **WdRulerStyle** behavior for center- and right-aligned tables can be unexpected; in these cases, the **SetLeftIndent** method should be used with care.

Example

This example creates a table in a new document and indents the first row 0.5 inch (36 points). When you change the left indent, the cell widths are adjusted to preserve the right edge of the table.

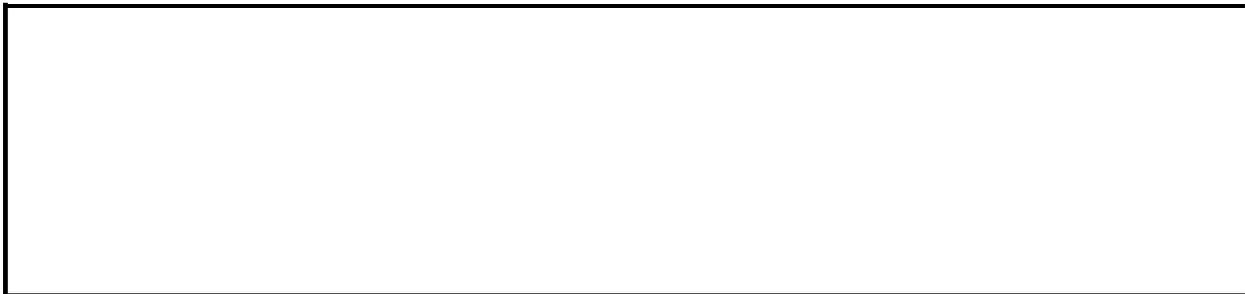
```
Dim docNew As Document
Dim tableNew As Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=3)

tableNew.Rows(1).SetLeftIndent LeftIndent:=InchesToPoints(0.5), _
    RulerStyle:=wdAdjustSameWidth
```

This example indents the first row in table one in the active document 18 points, and it narrows the width of the first column to preserve the position of the right edge of the table.

```
If ActiveDocument.Tables.Count >= 1 Then
    ActiveDocument.Tables(1).Rows.SetLeftIndent LeftIndent:=18, _
        RulerStyle:=wdAdjustFirstColumn
End If
```



SetLetterContent Method

-
Inserts the contents of the specified [LetterContent](#) object into a document.

expression.SetLetterContent(*LetterContent*)

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

LetterContent Required **LetterContent** object. The **LetterContent** object that includes the various elements of the letter.

Remarks

This method is similar to the [RunLetterWizard](#) method except that it doesn't display the Letter Wizard dialog box. The method adds, deletes, or restyles letter elements in the specified document based on the contents of the **LetterContent** object.

Example

This example retrieves the Letter Wizard elements from the active document, changes the attention line text, and then uses the **SetLetterContent** method to update the active document to reflect the changes.

```
Set myLetterContent = ActiveDocument.GetLetterContent  
myLetterContent.AttentionLine = "Greetings"  
ActiveDocument.SetLetterContent LetterContent:=myLetterContent
```



SetPasswordEncryptionOptions Method

Sets the options Microsoft Word uses for encrypting documents with passwords.

expression.**SetPasswordEncryptionOptions**(*PasswordEncryptionProvider*,
PasswordEncryptionAlgorithm, *PasswordEncryptionKeyLength*,
PasswordEncryptionFileProperties)

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

PasswordEncryptionProvider Required **String**. The name of the encryption provider.

PasswordEncryptionAlgorithm Required **String**. The name of the encryption algorithm. Word supports stream-encrypted algorithms.

PasswordEncryptionKeyLength Required **Long**. The encryption key length. Must be a multiple of 8, starting at 40.

PasswordEncryptionFileProperties Optional **Variant**. **True** for Word to encrypt file properties. Default is **True**.

Example

This example sets the password encryption options if the password encryption algorithm in use is "OfficeXor," which is the password algorithm used in versions of Word prior to Microsoft Word 97 for Windows.

```
Sub PasswordSettings()  
  With ActiveDocument  
    If .PasswordEncryptionAlgorithm = "OfficeXor" Then  
      .SetPasswordEncryptionOptions _  
        PasswordEncryptionProvider:="Microsoft RSA SChannel  
        PasswordEncryptionAlgorithm:="RC4", _  
        PasswordEncryptionKeyLength:=56, _  
        PasswordEncryptionFileProperties:=True  
    End If  
  End With  
End Sub
```



SetPosition Method

Sets the location of the node specified by ***Index***. Note that, 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 a **ShapeNodes** object.

Index Required **Long**. The node whose position is to be set.

X1, ***Y1*** Required **Single**. The position (in points) of the new node relative to the upper-left corner of the document.

Example

This example moves node two in the third shape on the active document to the right 200 points and down 300 points. The third shape must be a freeform drawing.

```
With ActiveDocument.Shapes(3).Nodes
  pointsArray = .Item(2).Points
  currXvalue = pointsArray(1, 1)
  currYvalue = pointsArray(1, 2)
  .SetPosition 2, currXvalue + 200, currYvalue + 300
End With
```



SetRange Method

Sets the starting and ending character positions for the range or selection.

Note Character position values start at the beginning of the story, with the first value being 0 (zero). All characters are counted, including nonprinting characters. Hidden characters are counted even if they're not displayed.

expression.**SetRange**(*Start*, *End*)

expression Required. An expression that returns a **Range** or **Selection** object.

Start Required **Long**. The starting character position of the range or selection.

End Required **Long**. The ending character position of the range or selection.

Remarks

The **SetRange** method redefines the starting and ending positions of an existing **Selection** or **Range** object. This method differs from the **Range** method, which is used to create a range, given a starting and ending position.

Example

This example selects the first 10 characters in the document.

```
Selection.SetRange Start:=0, End:=10
```

This example uses **SetRange** to redefine myRange to refer to the first three paragraphs in the active document.

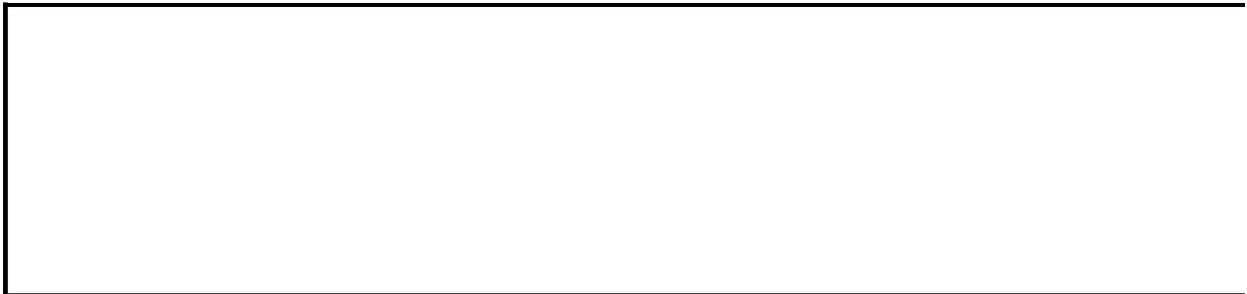
```
Set myRange = ActiveDocument.Paragraphs(1).Range  
myRange.SetRange Start:=myRange.Start, _  
    End:=ActiveDocument.Paragraphs(3).Range.End
```

This example uses **SetRange** to redefine myRange to refer to the area starting at the beginning of the document and ending at the end of the current selection.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)  
myRange.InsertAfter "Hello "  
myRange.SetRange Start:=myRange.Start, End:=Selection.End
```

This example extends the selection to the end of the document.

```
Selection.SetRange Start:=Selection.Start, _  
    End:=ActiveDocument.Content.End
```



SetSegmentType Method

Sets the segment type of the segment that follows the node specified by ***Index***. If the node is a control point for a curved segment, this method sets the segment type for that curve. Note that this may affect the total number of nodes by inserting or deleting adjacent nodes.

expression.**SetSegmentType**(***Index***, ***SegmentType***)

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

Index Required **Long**. The node whose segment type is to be set.

SegmentType Required [MsoSegmentType](#). Specifies if the segment is straight or curved.

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentLine

msoSegmentCurve

Example

This example changes all straight segments to curved segments in the third shape on the active document. The third shape must be a freeform drawing.

```
Dim lngLoop As Long
```

```
With ActiveDocument.Shapes(3).Nodes
```

```
    lngLoop = 1
```

```
    While lngLoop <= .Count
```

```
        If .Item(lngLoop).SegmentType = msoSegmentLine Then
```

```
            .SetSegmentType lngLoop, msoSegmentCurve
```

```
        End If
```

```
        lngLoop = lngLoop + 1
```

```
    Wend
```

```
End With
```



SetShapesDefaultProperties Method

-
Applies the formatting of the specified shape to a default shape for that document. New shapes inherit many of their attributes from the default shape.

expression.**SetShapesDefaultProperties**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Remarks

Using this method is equivalent to using the **Set AutoShape Defaults** command on the **Draw** menu on the **Drawing** toolbar.

Example

This example adds a rectangle to myDocument, 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.

```
Set mydocument = ActiveDocument
With mydocument.Shapes
  With .AddShape(msoShapeRectangle, 5, 5, 80, 60)
    With .Fill
      .ForeColor.RGB = RGB(0, 0, 255)
      .BackColor.RGB = RGB(0, 204, 255)
      .Patterned msoPatternHorizontalBrick
    End With
    ' Sets formatting for default shapes
    .SetShapesDefaultProperties
  End With
  ' New shape has default formatting
  .AddShape msoShapeRectangle, 90, 90, 40, 30
End With
```



SetThreeDFormat Method

Sets the preset extrusion format. Each preset extrusion format contains a set of preset values for the various properties of the extrusion.

expression.**SetThreeDFormat**(*PresetThreeDFormat*)

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

PresetThreeDFormat Required [MsoPresetThreeDFormat](#). Specifies a preset extrusion format that corresponds to one of the options (numbered from left to right, top to bottom) displayed when you click the **3-D** button on the **Drawing** toolbar.

MsoPresetThreeDFormat can be one of these MsoPresetThreeDFormat constants. Note that specifying **msoPresetThreeDFormatMixed** for this argument causes an error.

msoThreeD1

msoThreeD11

msoThreeD13

msoThreeD15

msoThreeD17

msoThreeD19

msoThreeD20

msoThreeD4

msoThreeD6

msoThreeD8

msoPresetThreeDFormatMixed

msoThreeD10

msoThreeD12

msoThreeD14

msoThreeD16

msoThreeD18
msoThreeD2
msoThreeD3
msoThreeD5
msoThreeD7
msoThreeD9

Remarks

This method sets the [PresetThreeDFormat](#) property to the format specified by the *PresetThreeDFormat* argument.

Example

This example adds an oval to the active document and sets its extrusion format to 3D Style 12.

```
With ActiveDocument.Shapes.AddShape(msoShapeOval, _  
    30, 30, 50, 25).ThreeD  
    .Visible = True  
    .SetThreeDFormat msoThreeD12  
End With
```



SetWidth Method

Sets the width of columns or cells in a table.

expression.**SetWidth**(*ColumnWidth*, *RulerStyle*)

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

ColumnWidth Required **Single**. The width of the specified column or columns, in points.

RulerStyle Required [WdRulerStyle](#). Controls the way Word adjusts cell widths.

WdRulerStyle can be one of these WdRulerStyle constants.

wdAdjustNone Sets the width of all selected cells or columns to the specified value. Word preserves the width of all non-selected columns, shifting them to the right or left as necessary. This is the default value.

wdAdjustSameWidth Sets the width of the cells in the first column only to the specified value. Word preserves the right edge of the table by adjusting the width of all other cells or columns to the same value.

wdAdjustFirstColumn Sets the width of the cells in the first column only to the specified value. If there is more than one column, Word preserves the right edge of the table and the positions of the other columns.

wdAdjustProportional Sets the width of the cells in the first column only to the specified value. If multiple columns are selected, Word preserves the right edge of the table and the positions of the non-selected columns by proportionally adjusting the width of the other selected columns. If only one cell or column is selected, Word preserves the right edge of the table by proportionally adjusting the width of the other cells or columns.

Remarks

The **WdRulerStyle** behavior described above applies to left-aligned tables. The **WdRulerStyle** behavior for center- and right-aligned tables can be unexpected; in these cases, the **SetWidth** method should be used with care.

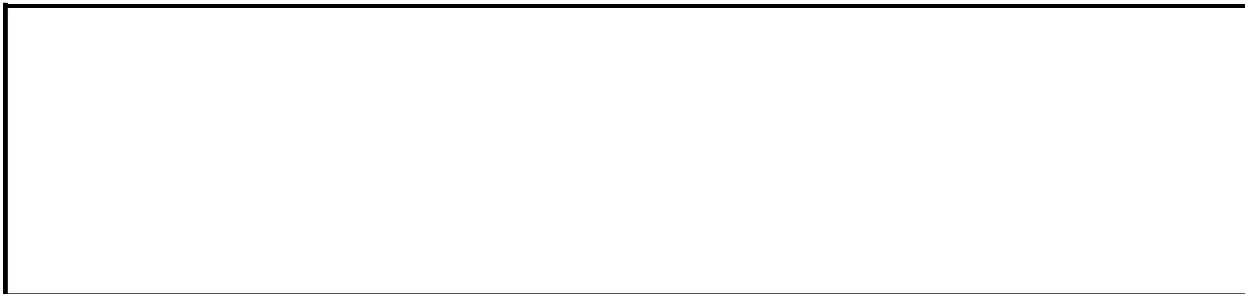
Example

This example creates a table in a new document and sets the width of the first cell in the second row to 1.5 inches. The example preserves the widths of the other cells in the table.

```
Set newDoc = Documents.Add
Set myTable = _
    newDoc.Tables.Add(Range:=Selection.Range, NumRows:=3, _
        NumColumns:=3)
myTable.Cell(2,1).SetWidth _
    ColumnWidth:=InchesToPoints(1.5), _
    RulerStyle:=wdAdjustNone
```

This example sets the width of the cell that contains the insertion point to 36 points. The example also narrows the first column to preserve the position of the right edge of the table.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells(1).SetWidth ColumnWidth:=36, _
        RulerStyle:=wdAdjustFirstColumn
Else
    MsgBox "The insertion point is not in a table."
End If
```



SetWPHelpOptions Method

Sets the options for the WordPerfect Help feature.

expression.**SetWPHelpOptions**(*CommandKeyHelp*, *DocNavigationKeys*, *MouseSimulation*, *DemoGuidance*, *DemoSpeed*, *HelpType*)

expression Required. An expression that returns an **Options** object.

CommandKeyHelp Optional **Variant**. **True** to display instructions or demonstrate a Word equivalent when a WordPerfect® for DOS key combination is pressed. WordPerfect Help is displayed in the status bar.

DocNavigationKeys Optional **Variant**. **True** to make the arrow keys and the PAGE UP, PAGE DOWN, HOME, END, and ESC keys function as they would in WordPerfect.

MouseSimulation Optional **Variant**. **True** to have the pointer move to each option that WordPerfect Help selects during demonstrations.

DemoGuidance Optional **Variant**. **True** to display help text when user input is required during WordPerfect Help demonstrations.

DemoSpeed Optional **Variant**. The speed of WordPerfect Help demonstrations. Can be one of the following values.

Value	Speed
0 (zero)	Fast
1	Medium
2	Slow

HelpType Optional **Variant**. Specifies whether WordPerfect Help displays help text or demonstrates the WordPerfect command. Can be either 0 (zero), for Help text, or 1, for a demonstration.

Example

This example sets the WordPerfect Help options.

```
Options.SetWPHelpOptions _  
    CommandKeyHelp:=True, _  
    DocNavigationKeys:=True, _  
    MouseSimulation:=True, _  
    DemoGuidance:=True, _  
    DemoSpeed:=0, _  
    HelpType:=0
```



Show Method

-

Displays and carries out actions initiated in the specified built-in Word dialog box. Returns a **Long** that indicates which button was clicked to close the dialog box.

Return value	Description
-2	The Close button.
-1	The OK button.
0 (zero)	The Cancel button.
> 0 (zero)	A command button: 1 is the first button, 2 is the second button, and so on.

Note Use the [Display](#) method to display a dialog box but not have any actions carried out or settings applied when the dialog box is closed.

expression.**Show**(*TimeOut*)

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

TimeOut Optional **Variant**. The amount of time that Word will wait before closing the dialog box automatically. One unit is approximately 0.001 second. Concurrent system activity may increase the effective time value. If this argument is omitted, the dialog box is closed when the user dismisses it.

Example

This example displays the **Find and Replace** dialog box with the word "Blue" preset in the **Find what** text box.

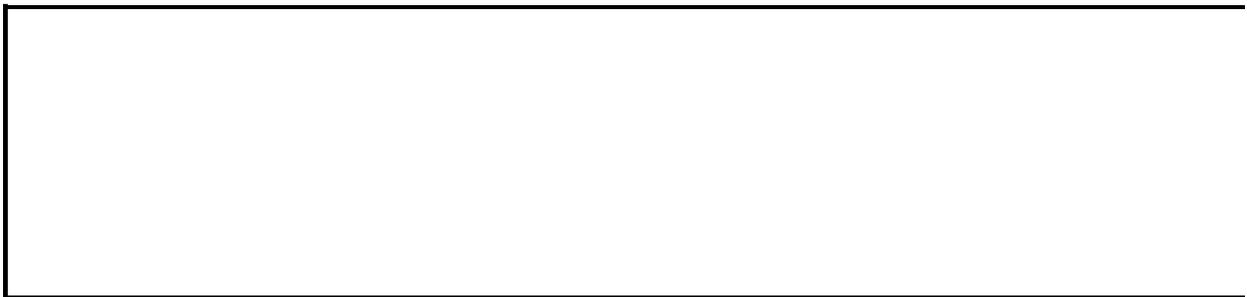
```
With Dialogs(wdDialogEditFind)
    .Find = "Blue"
    .Show
End With
```

This example displays and carries out any action initiated in the **Open** dialog box. The file name is set to *.* so that all file names are displayed.

```
With Dialogs(wdDialogFileOpen)
    .Name = *.*
    .Show
End With
```

This example displays and carries out any action initiated in the **Zoom** dialog box. If there are no actions initiated for approximately 9 seconds, the dialog box is closed.

```
Dialogs(wdDialogViewZoom).Show Timeout:=9000
```



ShowAllHeadings Method

-

Toggles between showing all text (headings and body text) and showing only headings.

Note This method generates an error if the view isn't outline view or master document view.

expression.**ShowAllHeadings**

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

Example

This example uses the **ShowHeading** method to show all headings (without any body text) and then toggles the display to show all text (headings and body text) in outline view.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdOutlineView  
    .ShowHeading 9  
    .ShowAllHeadings  
End With
```



ShowHeading Method

-

Shows all headings up to the specified heading level and hides subordinate headings and body text.

Note This method generates an error if the view isn't outline view or master document view.

expression.**ShowHeading**(*Level*)

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

Level Required **Long**. The outline heading level (a number from 1 to 9).

Example

This example switches the active window to outline view and displays all text that's formatted with the Heading 1 style. Body text and all other types of headings are hidden.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdOutlineView
    .ShowHeading 1
End With
```

This example switches the window for Document1 to outline view and displays all text that's formatted with the Heading 1, Heading 2, or Heading 3 style.

```
With Windows("Document1").View
    .Type = wdOutlineView
    .ShowHeading 3
End With
```



ShowMe Method

-

Displays the Office Assistant or the Help window when there's more information available. If additional information isn't available, this method generates a message that no associated Help topic exists.

expression.**ShowMe()**

expression An expression that returns an **Application** object.

Example

This examples completes a TipWizard Show Me action if one's available.

Application.**ShowMe**



ShowWizard Method

Displays the Mail Merge Wizard in a document.

expression.**ShowWizard**(*InitialState*, *ShowDocumentStep*, *ShowTemplateStep*, *ShowDataStep*, *ShowWriteStep*, *ShowPreviewStep*, *ShowMergeStep*)

expression Required. An expression that returns a [MailMerge](#) object.

InitialState Required **Variant**. The number of the Mail Merge Wizard step to display.

ShowDocumentStep Optional **Variant**. **True** keeps the "Select document type" step in the sequence of mail merge steps. **False** removes step one.

ShowTemplateStep Optional **Variant**. **True** keeps the "Select starting document" step in the sequence of mail merge steps. **False** removes step two.

ShowDataStep Optional **Variant**. **True** keeps the "Select recipients" step in the sequence of mail merge steps. **False** removes step three.

ShowWriteStep Optional **Variant**. **True** keeps the "Write your letter" step in the sequence of mail merge steps. **False** removes step four.

ShowPreviewStep Optional **Variant**. **True** keeps the "Preview your letters" step in the sequence of mail merge steps. **False** removes step five.

ShowMergeStep Optional **Variant**. **True** keeps the "Complete the merge" step in the sequence of mail merge steps. **False** removes step six.

Example

This example checks if the Mail Merge Wizard is already displayed and, if it is, moves to the Mail Merge Wizard's sixth step and removes the fifth step from the Wizard.

```
Sub ShowMergeWizard()  
  With ActiveDocument.MailMerge  
    If .WizardState > 0 Then  
      .ShowWizard InitialState:=6, ShowPreviewStep:=False  
    End If  
  End With  
End Sub
```



↳ [Show All](#)

Shrink Method

-

Font object: 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.

Selection object: Shrinks the selection to the next smaller unit of text. The progression is as follows: entire document, section, paragraph, sentence, word, insertion point.

expression.**Shrink**

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

Example

▶ [As it applies to the **Font** object.](#)

This example inserts a line of increasingly smaller Z's in a new document.

```
Set myRange = Documents.Add.Content
myRange.Font.Size = 45
For Count = 1 To 5
    myRange.InsertAfter "Z"
    For Count2 = 1 to 3
        myRange.Characters(Count).Font.Shrink
    Next Count2
Next Count
```

▶ [As it applies to the **Selection** object.](#)

This example reduces the font size of the selected text by one size.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Shrink
Else
    MsgBox "You need to select some text."
End If
```



ShrinkDiscontiguousSelection Method

-
Deselects all but the most recently selected text when a selection contains multiple, unconnected selections.

expression.**ShrinkDiscontiguousSelection**

expression Required. An expression that returns a [Selection](#) object.

Example

This example deselects all but the most recently selected text and formats with bold and small caps the text remaining in the selection. This example assumes there are multiple selections in the document.

```
Sub ShrinkMultipleSelection()  
  With Selection  
    .ShrinkDiscontiguousSelection  
    .Font.Bold = True  
    .Font.SmallCaps = True  
  End With  
End Sub
```



SmallScroll Method

Scrolls a window or pane by the specified number of lines. This method is equivalent to clicking the scroll arrows on the horizontal and vertical scroll bars.

expression.SmallScroll(Down, Up, ToRight,ToLeft)

expression Required. An expression that returns a **Pane** or **Window** object.

Down Optional **Variant**. The number of lines to scroll the window down. A "line" corresponds to the distance scrolled by clicking the down scroll arrow on the vertical scroll bar once.

Up Optional **Variant**. The number of lines to scroll the window up. A "line" corresponds to the distance scrolled by clicking the up scroll arrow on the vertical scroll bar once.

ToRight Optional **Variant**. The number of lines to scroll the window to the right. A "line" corresponds to the distance scrolled by clicking the right scroll arrow on the horizontal scroll bar once.

ToLeft Optional **Variant**. The number of lines to scroll the window to the left. A "line" corresponds to the distance scrolled by clicking the left scroll arrow on the horizontal scroll bar once.

Remarks

If ***Down*** and ***Up*** are both specified, the window is scrolled by the difference of the arguments. For example, if ***Down*** is 3 and ***Up*** is 6, the window is scrolled up three lines. Similarly, if ***ToLeft*** and ***ToRight*** are both specified, the window is scrolled by the difference of the arguments.

Any of these arguments can be a negative number. If no arguments are specified, the window is scrolled down by one line.

Example

This example scrolls the active window down one line.

```
ActiveDocument.ActiveWindow.SmallScroll Down:=1
```

This example splits the active window and then scrolls up and over to the left.

```
With ActiveDocument.ActiveWindow  
    .Split = True  
    .SmallScroll Up:=5,ToLeft:=5  
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 a **FillFormat** object.

Example

This example converts all fills on the active document to uniform red fills.

```
Dim shapeLoop As Shape
```

```
For Each shapeLoop In ActiveDocument.Shapes
```

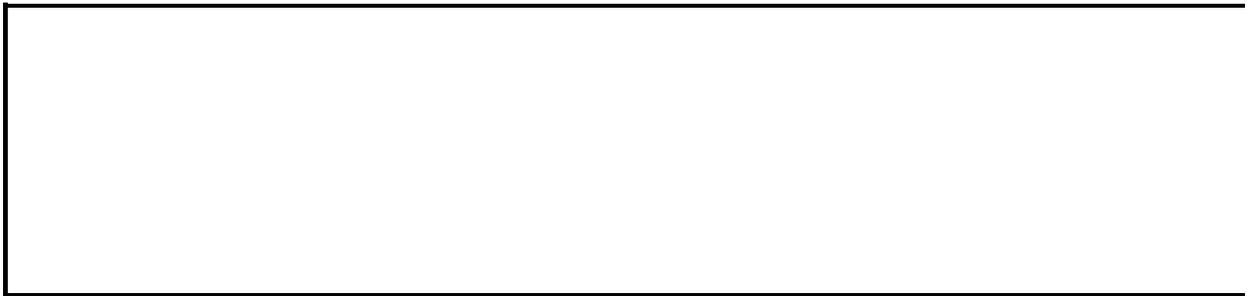
```
  With shapeLoop.Fill
```

```
    .Solid
```

```
    .ForeColor.RGB = RGB(255, 0, 0)
```

```
  End With
```

```
Next
```



↳ [Show All](#)

Sort Method

▶ [Sort method as it applies to the **Column** object.](#)

Sorts the specified table column.

expression.Sort(ExcludeHeader, SortFieldType, SortOrder, CaseSensitive, BidiSort, IgnoreThe, IgnoreKashida, IgnoreDiacritics, IgnoreHe, LanguageID)

expression Required. An expression that returns a [Column](#) object.

ExcludeHeader Optional **Variant**. **True** to exclude the first row or paragraph header from the sort operation. The default value is **False**.

SortFieldType Optional **Variant**. The sort type for the column. Can be one of the [WdSortFieldType](#) constants.

wdSortFieldAlphanumeric *Default*

wdSortFieldDate

wdSortFieldJapanJIS

wdSortFieldKoreaKS

wdSortFieldNumeric

wdSortFieldStroke

wdSortFieldSyllable

SortOrder Optional **Variant**. The sorting order to use for the column. Can be one [WdSortOrder](#) constant.

wdSortOrderAscending *Default*

wdSortOrderDescending

CaseSensitive Optional **Variant**. **True** to sort with case sensitivity. The default value is **False**.

BidiSort Optional **Variant**. **True** to sort based on right-to-left language rules. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreThe Optional **Variant**. **True** to ignore the Arabic character *alef lam* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreKashida Optional **Variant**. **True** to ignore kashidas when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreDiacritics Optional **Variant**. **True** to ignore bidirectional control characters when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreHe Optional **Variant**. **True** to ignore the Hebrew character *he* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

LanguageID Optional **Variant**. **LanguageID** Optional **Variant**. Specifies the sorting language. Can be one of the **WdLanguageID** constants. Refer to the Object Browser for a list of the [WdLanguageID](#) constants.

Remarks

If you want to sort paragraphs within a table cell, include only the paragraphs and not the end-of-cell mark; if you include the end-of-cell mark in a selection or range and then attempt to sort the paragraphs, Word displays a message stating that it found no valid records to sort.

► [Sort method as it applies to the **Range** and **Selection** objects.](#)

Sorts the paragraphs in the specified range or selection.

expression.Sort(ExcludeHeader, FieldNumber, SortFieldType, SortOrder, FieldNumber2, SortFieldType2, SortOrder2, FieldNumber3, SortFieldType3, SortOrder3, SortColumn, Separator, CaseSensitive, BidiSort, IgnoreThe, IgnoreKashida, IgnoreDiacritics, IgnoreHe, LanguageID)

expression Required. An expression that returns one of the above objects.

ExcludeHeader Optional **Variant**. **True** to exclude the first row or paragraph header from the sort operation. The default value is **False**.

FieldNumber, FieldNumber2, FieldNumber3 Optional **Variant**. The fields to sort by. Microsoft Word sorts by **FieldNumber**, then by **FieldNumber2**, and then by **FieldNumber3**.

SortFieldType, SortFieldType2, SortFieldType3 Optional **Variant**. The respective sort types for **FieldNumber, FieldNumber2, and FieldNumber3**. Can be one of the [WdSortFieldType](#) constants.

wdSortFieldAlphanumeric

wdSortFieldDate

wdSortFieldJapanJIS

wdSortFieldKoreaKS

wdSortFieldNumeric

wdSortFieldStroke

wdSortFieldSyllable

The default value is **wdSortFieldAlphanumeric**. Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

SortOrder, SortOrder2, SortOrder3 Optional **VARIANT**. The sorting order to use when sorting *FieldNumber*, *FieldNumber2*, and *FieldNumber3*. Can be one [WdSortOrder](#) constant.

wdSortOrderAscending Default.

wdSortOrderDescending

SortColumn Optional **VARIANT**. **True** to sort only the column specified by the **Range** or **Selection** object.

Separator Optional **VARIANT**. The type of field separator. Can be one of the [WdSortSeparator](#) constants.

wdSortSeparateByCommas Default.

wdSortSeparateByDefaultTableSeparator

wdSortSeparateByTabs

CaseSensitive Optional **VARIANT**. **True** to sort with case sensitivity. The default value is **False**.

BidiSort Optional **VARIANT**. **True** to sort based on right-to-left language rules. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreThe Optional **VARIANT**. **True** to ignore the Arabic character *alef lam* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreKashida Optional **VARIANT**. **True** to ignore kashidas when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreDiacritics Optional **VARIANT**. **True** to ignore bidirectional control

characters when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreHe Optional **Variant**. **True** to ignore the Hebrew character *he* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

LanguageID Optional **Variant**. **LanguageID** Optional **Variant**. Specifies the sorting language. Can be one of the **WdLanguageID** constants. Refer to the Object Browser for a list of the [WdLanguageID](#) constants.

SubFieldNumber, SubFieldNumber2, SubFieldNumber3 Optional **Variant**. (Applies to the Selection object only.)

► [Sort method as it applies to the Table object.](#)

Sorts the specified table.

expression.Sort(ExcludeHeader, FieldNumber, SortFieldType, SortOrder, FieldNumber2, SortFieldType2, SortOrder2, FieldNumber3, SortFieldType3, SortOrder3, CaseSensitive, BidiSort, IgnoreThe, IgnoreKashida, IgnoreDiacritics, IgnoreHe, LanguageID)

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

ExcludeHeader Optional **Variant**. **True** to exclude the first row or paragraph header from the sort operation. The default value is **False**.

FieldNumber, FieldNumber2, FieldNumber3 Optional **Variant**. The fields to sort by. Microsoft Word sorts by **FieldNumber**, then by **FieldNumber2**, and then by **FieldNumber3**.

wdSortFieldAlphanumeric

wdSortFieldDate

wdSortFieldJapanJIS

wdSortFieldKoreaKS

wdSortFieldNumeric

wdSortFieldStroke **wdSortFieldSyllable**

The default value is **wdSortFieldAlphanumeric**. Some of these constants may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

SortOrder, SortOrder2, SortOrder3 Optional **Variant**. The sorting order to use when sorting *FieldNumber*, *FieldNumber2*, and *FieldNumber3*. Can be one [WdSortOrder](#) constant.

wdSortOrderAscending Default.
wdSortOrderDescending

CaseSensitive Optional **Variant**. **True** to sort with case sensitivity. The default value is **False**.

BidiSort Optional **Variant**. **True** to sort based on right-to-left language rules. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreThe Optional **Variant**. **True** to ignore the Arabic character *alef lam* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreKashida Optional **Variant**. **True** to ignore kashidas when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreDiacritics Optional **Variant**. **True** to ignore bidirectional control characters when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

IgnoreHe Optional **Variant**. **True** to ignore the Hebrew character *he* when sorting right-to-left language text. This argument may not be available to you, depending on the language support (U.S. English, for example) that you've

selected or installed.

LanguageID Optional **Variant**. Specifies the sorting language. Can be one of the **WdLanguageID** constants. Refer to the Object Browser for a list of the [WdLanguageID](#) constants.

Example

▶ [As it applies to the **Table** object.](#)

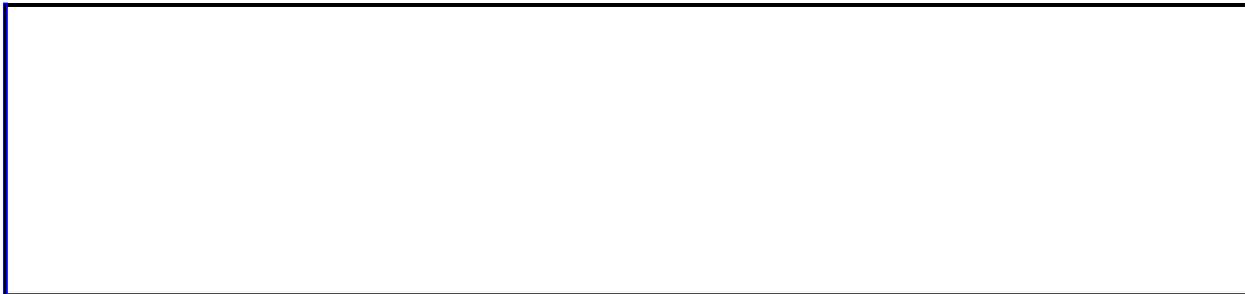
This example sorts the first table in the active document, excluding the heading row.

```
Sub NewTableSort()  
    ActiveDocument.Tables(Index:=1)  
    Selection.Sort ExcludeHeader:=True  
End Sub
```

▶ [As it applies to the **Range** or **Selection** object.](#)

This example inserts three lines of text into a new document and then sorts the lines in ascending alphanumeric order

```
Sub NewParagraphSort()  
    Dim newDoc As Document  
    Set newDoc = Documents.Add  
    newDoc.Content.InsertAfter "pear" & Chr(13) _  
        & "zucchini" & Chr(13) & "apple" & Chr(13)  
    newDoc.Content.Sort SortOrder:=wdSortOrderAscending  
End Sub
```



SortAscending Method

-

Sorts paragraphs or table rows in ascending alphanumeric order. The first paragraph or table row is considered a header record and isn't included in the sort. Use the [Sort](#) method to include the header record in a sort.

Note This method offers a simplified form of sorting intended for mail merge data sources that contain columns of data. For most sorting tasks, use the [Sort](#) method.

expression.**SortAscending**

expression Required. An expression that returns a **Range**, **Selection**, or **Table** object.

Example

This example sorts the table that contains the selection in ascending order.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).SortAscending
Else
    MsgBox "The insertion point is not in a table."
End If
```



SortDescending Method

-

Sorts paragraphs or table rows in descending alphanumeric order. The first paragraph or table row is considered a header record and isn't included in the sort. Use the [Sort](#) method to include the header record in a sort.

Note This method offers a simplified form of sorting intended for mail-merge data sources that contain columns of data. For most sorting tasks, use the [Sort](#) method.

expression.**SortDescending**

expression Required. An expression that returns a **Range**, **Selection**, or **Table** object.

Example

This example creates a 5x5 table in a new document, inserts text into each cell, and then sorts the table in descending alphanumeric order.

```
Set newDoc = Documents.Add
Set myTable = _
    newDoc.Tables.Add(Range:=Selection.Range, NumRows:=5, _
        NumColumns:=5)
For iRow = 1 To myTable.Rows.Count
    For iCol = 1 To myTable.Columns.Count
        Set MyRange = myTable.Rows(iRow).Cells(iCol).Range
        MyRange.InsertAfter "Cell" & Str$(iRow) & "," & Str$(iCol)
    Next iCol
Next iRow
MsgBox "Click OK to sort in descending order."
myTable.SortDescending
```

This example sorts the table that contains the insertion point in descending alphanumeric order.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).SortDescending
Else
    MsgBox "The insertion point is not in a table."
End If
```



Space1 Method

-

Single-spaces the specified paragraphs. The exact spacing is determined by the font size of the largest characters in each paragraph.

expression.**Space1**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Remarks

The following two statements are equivalent:

```
ActiveDocument.Paragraphs(1).Space1
```

```
ActiveDocument.Paragraphs(1).LineSpacingRule = wdLineSpaceSingle
```

Example

This example changes the first paragraph in the active document to single spacing.

ActiveDocument.Paragraphs(1).Space1



Space15 Method

-

Formats the specified paragraphs with 1.5-line spacing. The exact spacing is determined by adding 6 points to the font size of the largest character in each paragraph.

expression.**Space15**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Remarks

The following two statements are equivalent:

```
ActiveDocument.Paragraphs(1).Space15
```

```
ActiveDocument.Paragraphs(1).LineSpacingRule = wdLineSpace1pt5
```

Example

This example changes the first paragraph in the active document to 1.5-line spacing.

```
ActiveDocument.Paragraphs(1).Space15
```



Space2 Method

-

Double-spaces the specified paragraphs. The exact spacing is determined by adding 12 points to the font size of the largest character in each paragraph.

expression.**Space2**

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Remarks

The following two statements are equivalent:

```
ActiveDocument.Paragraphs(1).Space2
```

```
ActiveDocument.Paragraphs(1).LineSpacingRule = wdLineSpaceDouble
```

Example

This example changes the first paragraph in the selection to double spacing.

`Selection.Paragraphs(1).Space2`



↳ [Show All](#)

Split Method

▶ [Split method as it applies to the **Cell** object.](#)

Splits a single table cell into multiple cells.

expression.**Split**(*NumRows*, *NumColumns*)

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

NumRows Optional **Variant**. The number of rows that the cell or group of cells is to be split into.

NumColumns Optional **Variant**. The number of columns that the cell or group of cells is to be split into.

▶ [Split method as it applies to the **Cells** object.](#)

Splits a range of table cells.

expression.**Split**(*NumRows*, *NumColumns*, *MergeBeforeSplit*)

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

NumRows Optional **Variant**. The number of rows that the cell or group of cells is to be split into.

NumColumns Optional **Variant**. The number of columns that the cell or group of cells is to be split into.

MergeBeforeSplit Optional **Variant**. **True** to merge the cells with one another before splitting them.

▶ [Split method as it applies to the **Subdocument** object.](#)

Divides an existing subdocument into two subdocuments at the same level in

master document view or outline view. The division is at the beginning of the specified range. If the active document isn't in either master document or outline view, or if the range isn't at the beginning of a paragraph in a subdocument, an error occurs.

expression.**Split**(*Range*)

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

Range Required **Range** object. The range that, when the subdocument is split, becomes a separate subdocument.

▶ [Split method as it applies to the **Table** object.](#)

Inserts an empty paragraph immediately above the specified row in the table, and returns a **Table** object that contains both the specified row and the rows that follow it.

expression.**Split**(*BeforeRow*)

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

BeforeRow Required **Variant**. The row that the table is to be split before. Can be a row number or a **Row** object.

Example

▶ [As it applies to the **Cell** object.](#)

This example splits the first cell in the first table into two cells.

```
ActiveDocument.Tables(1).Cell(1, 1).Split NumColumns:=2
```

▶ [As it applies to the **Cells** object.](#)

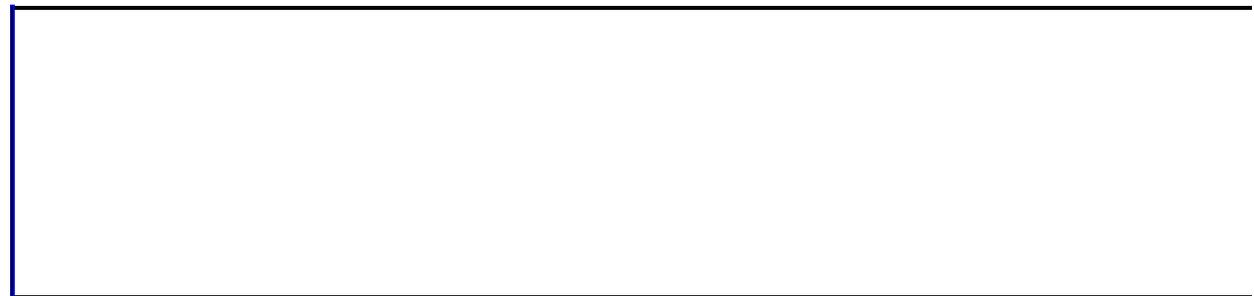
This example merges the selected cells into a single cell and then splits the cell into three cells in the same row.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.Cells.Split NumRows:=1, NumColumns:=3, _  
        MergeBeforeSplit:= True  
End If
```

▶ [As it applies to the **Subdocument** object.](#)

This example splits the selection from an existing subdocument into a separate subdocument.

```
Selection.Range.Subdocuments(1).Split Range:=Selection.Range
```



▶ [As it applies to the **Table** object.](#)

This example creates a 5x5 table in the active document and splits it before the third row. Shading is applied to the cells in the resulting table (the new 3x5 table).

```
Set newDoc = Documents.Add  
Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _
```

```
NumColumns:=5, NumRows:=5)  
myTable.Split(BeforeRow:=myTable.Rows(3)).Shading _  
.Texture = wdTexture10Percent
```

SplitTable Method

-

Inserts an empty paragraph above the first row in the selection. If the selection isn't in the first row of the table, the table is split into two tables.

Note If the selection isn't in a table, an error occurs.

expression.**SplitTable**

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

Example

If the selection is in a table, this example splits the table.

```
If Selection.Information(wdWithInTable) = True Then  
    Selection.SplitTable  
End If
```

This example splits the first table in the active document between the first and second rows.

```
ActiveDocument.Tables(1).Rows(2).Select  
Selection.SplitTable
```



↳ [Show All](#)

StartOf Method

Moves or extends the start position of the specified range or selection to the beginning of the nearest specified text unit. This method returns a **Long** that indicates the number of characters by which the range or selection was moved or extended. The method returns a negative number if the movement is backward through the document.

expression.**StartOf**(*Unit*, *Extend*)

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

Unit Optional [WdUnits](#). The unit by which the start position of the specified range or selection is to be moved.

WdUnits can be one of these WdUnits constants.

wdCell

wdCharacter

wdColumn

wdParagraph

wdRow

wdSection

wdSentence

wdStory

wdTable

wdWord

If *expression* returns a **Selection** object, you can also use **wdLine**. The default value is **wdWord**.

Extend Optional [WdMovement](#).

WdMovementType can be one of these WdMovementType constants.

wdMove

wdExtend

If you use **wdMove**, both ends of the range or selection are moved to the beginning of the specified unit. If you use **wdExtend**, the beginning of the range or selection is extended to the beginning of the specified unit. The default value is **wdMove**.

Remarks

If the beginning of the specified range or selection is already at the beginning of the specified unit, this method doesn't move or extend the range or selection. For example, if the selection is at the beginning of a line, the following example returns 0 (zero) and doesn't change the selection.

```
char = Selection.StartOf(Unit:=wdLine, Extend:=wdMove)
```

Example

This example selects the text from the insertion point to the beginning of the line. The number of characters selected is stored in charmoved.

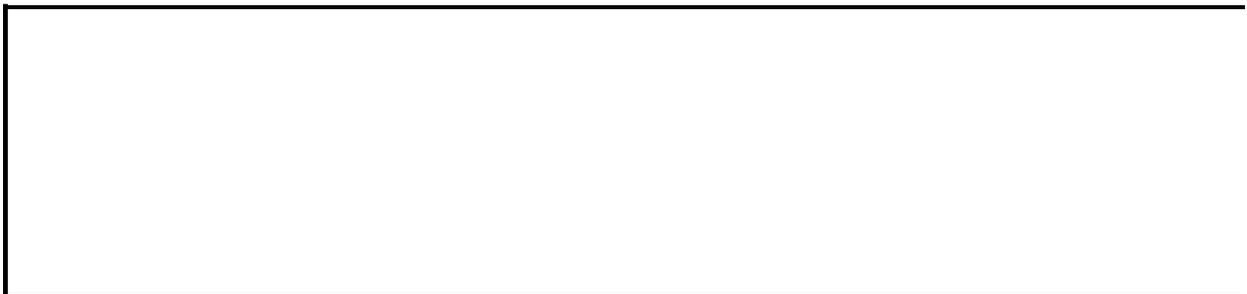
```
Selection.Collapse Direction:=wdCollapseStart charmoved = Selection.
```

This example moves the selection to the beginning of the paragraph.

```
Selection.StartOf Unit:=wdParagraph, Extend:=wdMove
```

This example moves myRange to the beginning of the second sentence in the document (myRange is collapsed and positioned at the beginning of the second sentence). The example uses the **Select** method to show the location of myRange.

```
Set myRange = ActiveDocument.Sentences(2)  
myRange.StartOf Unit:=wdSentence, Extend:=wdMove  
myRange.Select
```



SubstituteFont Method

Sets font-mapping options, which are reflected in the **Font Substitution** dialog box (**Compatibility** tab, **Options** dialog box, **Tools** menu).

expression.**SubstituteFont**(*UnavailableFont*, *SubstituteFont*)

expression Required. An expression that returns an **Application** object.

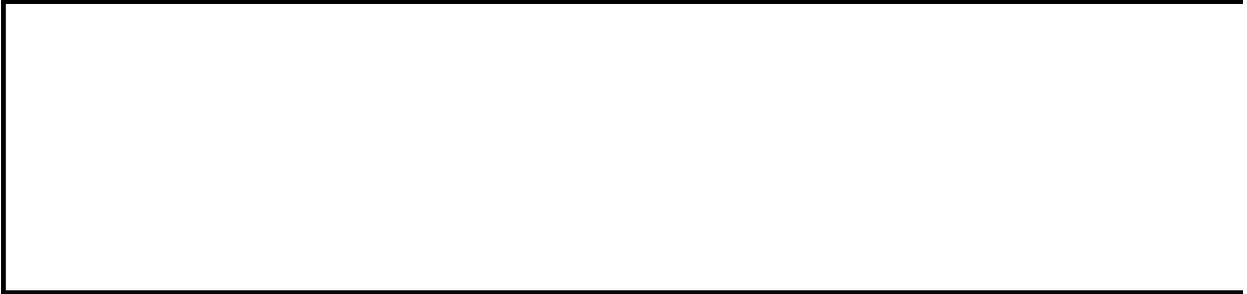
UnavailableFont Required **String**. The name of a font not available on your computer that you want to map to a different font for display and printing.

SubstituteFont Required **String**. The name of a font available on your computer that you want to substitute for the unavailable font.

Example

This example substitutes Courier for CustomFont1.

```
Application.SubstituteFont UnavailableFont:= "CustomFont1", _  
    SubstituteFont:= "Courier"
```



SwapNode Method

-

Swaps the target diagram node with the source diagram node. Any child diagram nodes are moved along with their corresponding root nodes.

expression.**SwapNode**(*TargetNode*)

expression Required. An expression that returns a [DiagramNode](#) object.

TargetNode Required **DiagramNode** object. The node with which to swap.

Example

The following example swaps two nodes in a newly-created diagram.

```
Sub SwapNode()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Object  
    Dim intCount As Integer  
  
    'Add organizational chart to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add first node to organizational chart  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes to the first node  
    For intCount = 1 To 3  
        dgnNode.Children.AddNode  
    Next intCount  
  
    'Add three child nodes to the first child node  
    'of the first node  
    For intCount = 1 To 3  
        dgnNode.Children.Item(1).Children.AddNode  
    Next intCount  
  
    'Swap the first and third child nodes that were just created  
    dgnNode.Children.Item(1).SwapNode _  
        TargetNode:=dgnNode.Children.Item(3)  
End Sub
```



SwapWithEndnotes Method

-
Converts all footnotes in a document to endnotes and vice versa.

Note To convert a range of footnotes to endnotes, use the **Convert** method.

expression.**SwapWithEndnotes**

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

Example

This example converts the footnotes in the active document to endnotes and converts the endnotes to footnotes.

`ActiveDocument.Footnotes.SwapWithEndnotes`



SwapWithFootnotes Method

-
Converts all endnotes in a document to footnotes and vice versa.

Note To convert a range of endnotes to footnotes, use the **Convert** method.

expression.**SwapWithFootnotes**

expression Required. An expression that returns an **Endnotes** object.

Example

This example converts the endnotes in the active document to footnotes and converts the footnotes to endnotes.

```
ActiveDocument.Endnotes.SwapWithFootnotes
```



TabHangingIndent Method

-

Sets a hanging indent to a specified number of tab stops. Can be used to remove tab stops from a hanging indent if the value of **Count** is a negative number.

expression.**TabHangingIndent**(**Count**)

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Count Required **Integer**. The number of tab stops to indent (if positive) or the number of tab stops to remove from the indent (if negative).

Example

This example sets a hanging indent to the second tab stop for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).TabHangingIndent(2)
```

This example moves the hanging indent back one tab stop for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).TabHangingIndent(-1)
```



TabIndent Method

-

Sets the left indent for the specified paragraphs to a specified number of tab stops. Can also be used to remove the indent if the value of **Count** is a negative number.

expression.**TabIndent**(*Count*)

expression Required. An expression that returns a **Paragraph**, **Paragraphs**, or **ParagraphFormat** object.

Count Required **Integer**. The number of tab stops to indent (if positive) or the number of tab stops to remove from the indent (if negative).

Example

This example indents the first paragraph in the active document to the second tab stop.

```
ActiveDocument.Paragraphs(1).TabIndent(2)
```

This example moves the indent of the first paragraph in the active document back one tab stop.

```
ActiveDocument.Paragraphs(1).TabIndent(-1)
```



↳ [Show All](#)

TCSCConverter Method

Converts the specified range from Traditional Chinese to Simplified Chinese or vice versa.

expression.TCSCConverter(**WdTCSCConverterDirection**, **CommonTerms**, **UseVariants**)

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

WdTCSCConverterDirection Optional [WdTCSCConverterDirection](#).

WdTCSCConverterDirection can be one of these WdTCSCConverterDirection constants.

wdTCSCConverterDirectionAuto *default* Converts in the appropriate direction based on the detected language of the specified range.

wdTCSCConverterDirectionSCTC Converts from Simplified Chinese to Traditional Chinese.

wdTCSCConverterDirectionTCSC Converts from Traditional Chinese to Simplified Chinese.

CommonTerms Optional **Boolean**. **True** if Microsoft Word converts common expressions intact rather than converting on a character-by-character basis.

UseVariants Optional **Boolean**. **True** if Word uses Taiwan, Hong Kong SAR, and Macao character variants. Can only be used if translating from Simplified Chinese to Traditional Chinese.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example converts the current selection from Simplified Chinese to Traditional Chinese. It converts common expressions intact and uses regional character variants.

```
Selection.Range.TCSCConverter _  
    wdTCSCConverterDirectionSCTC, True, True
```



TOCInFrameset Method

-

Creates a table of contents based on the specified document and puts it in a new frame on the left side of the frames page.

expression.**TOCInFrameset**

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

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example opens a file named "Proposal.doc", creates a frames page based on the file, and adds a frame (on the left side of the page) containing a table of contents for the file.

```
Documents.Open "C:\Documents\Proposal.doc"  
ActiveDocument.ActiveWindow.ActivePane.NewFrameset  
ActiveDocument.ActiveWindow.ActivePane.TOCInFrameset
```



ToggleCharCode Method

-
Switches a selection between a Unicode character and its corresponding hexadecimal value.

expression.**ToggleCharCode**

expression Required. An expression that returns a [Selection](#) object.

Example

This example enters the hex value "20ac" at the cursor position and toggles that value to its corresponding Unicode character.

```
Sub ToggleCharCase()  
    Selection.TypeText Text:="20ac"  
    Selection.ToggleCharacterCode  
End Sub
```



ToggleFormsDesign Method

-

Toggles form design mode on or off. When Word is in form design mode, the **Control Toolbox** toolbar is displayed. You can use this toolbar to insert ActiveX controls such as command buttons, scroll bars, and option buttons. In form design mode, event procedures don't run, and when you click an embedded control, the control's sizing handles appear.

expression.**ToggleFormsDesign**

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

Example

This example switches to form design mode if the **Control Toolbox** toolbar isn't currently displayed.

```
If CommandBars("Control Toolbox").Visible = False Then  
    ActiveDocument.ToggleFormsDesign  
End If
```



ToggleHeader Method

-
Toggles the display of the header in the active e-mail message.

expression.**ToggleHeader**

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

Example

This example toggles the display of the header in the active e-mail message.

`Application.MailMessage.ToggleHeader`



ToggleKeyboard Method

-

Switches the keyboard language setting between right-to-left and left-to-right languages.

expression.**ToggleKeyboard**

expression Required. An expression that returns an [Application](#) object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example asks the user whether to switch the keyboard language setting between right-to-left and left-to-right languages.

```
x = MsgBox("Switch the keyboard language setting?", vbYesNo)
If x = vbYes Then Application.ToggleKeyboard
```



TogglePortrait Method

-

Switches between portrait and landscape page orientations for a document or section.

expression.**TogglePortrait**

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

Remarks

If the specified sections have different page orientations, an error occurs.

Example

This example changes the page orientation for the active document.

```
ActiveDocument.PageSetup.TogglePortrait
```

This example changes the page orientation for all the sections in the selection. If the initial orientation of each section is not the same as the orientation of the other sections, an error occurs.

```
Selection.PageSetup.TogglePortrait
```



ToggleShowCodes Method

Toggles the display of the fields between field codes and field results.

Note Use the [ShowCodes](#) property to control the display of an individual field.

expression.**ToggleShowCodes**

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

Example

This example toggles the display of fields in the selection (the equivalent of pressing SHIFT+F9).

`Selection.Fields.ToggleShowCodes`

This example toggles the display of fields in the active document (the equivalent of pressing ALT+F9).

`ActiveDocument.Fields.ToggleShowCodes`



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 a **TextEffectFormat** object.

Remarks

Using the **ToggleVerticalText** method swaps the values of the **Width** and **Height** properties of the **Shape** object that represents the WordArt and leaves the **Left** and **Top** 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 document and switches from horizontal text flow (the default for the specified WordArt style, **msoTextEffect1**) to vertical text flow.

```
Dim newWordArt As Shape
```

```
Set newWordArt = _  
    ActiveDocument.Shapes.AddTextEffect( _  
        PresetTextEffect:=msoTextEffect1, Text:="Test", _  
        FontName:="Arial Black", FontSize:=36, FontBold:=False, _  
        FontItalic:=False, Left:=100, Top:=100)  
newWordArt.TextEffect.ToggleVerticalText
```



TransferChildren Method

Moves the child nodes of the source diagram node to the target (receiving) diagram node.

expression.**TransferChildren**(*ReceivingNode*)

expression Required. An expression that returns a [DiagramNode](#) object.

ReceivingNode Required **DiagramNode** object. The node to which to transfer the child nodes.

Example

The following example transfers the child nodes of a newly-created diagram from one node to another.

```
Sub TransferChildNodes()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add organizational chart to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add first node to organizational chart  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes to first node  
    For intCount = 1 To 3  
        dgnNode.Children.AddNode  
    Next intCount  
  
    'Add three child nodes to the first child node  
    'of the first node  
    For intCount = 1 To 3  
        dgnNode.Children.Item(1).Children.AddNode  
    Next intCount  
  
    'Move the child nodes of the first child node  
    'so they become child nodes of the third child node  
    dgnNode.Children.Item(1).TransferChildren _  
        ReceivingNode:=dgnNode.Children.Item(3)  
  
End Sub
```



TwoColorGradient Method

Sets the specified fill to a two-color gradient.

expression.**TwoColorGradient**(*Style*, *Variant*)

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

Style Required [MsoGradientStyle](#). The gradient style.

MsoGradientStyle can be one of these MsoGradientStyle constants.

msoGradientDiagonalDown

msoGradientDiagonalUp

msoGradientFromCenter

msoGradientFromCorner

msoGradientFromTitle Used only in Microsoft PowerPoint.

msoGradientHorizontal

msoGradientMixed

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 **msoGradientFromCenter**, this argument can be either 1 or 2.

Example

This example adds a rectangle with a two-color gradient fill to the active document and sets the background and foreground color for the fill.

```
With ActiveDocument.Shapes.AddShape(msoShapeRectangle, _  
    0, 0, 40, 80).Fill  
    .ForeColor.RGB = RGB(128, 0, 0)  
    .BackColor.RGB = RGB(0, 170, 170)  
    .TwoColorGradient msoGradientHorizontal, 1  
End With
```



TypeBackspace Method

-

Deletes the character preceding a collapsed selection (an insertion point). If the selection isn't collapsed to an insertion point, the selection is deleted.

Note This method corresponds to functionality of the BACKSPACE key.

expression.**TypeBackspace**

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

Example

This example deletes the character preceding the insertion point (the collapsed selection).

```
With Selection
    .Collapse Direction:=wdCollapseEnd
    .TypeBackspace
End With
```

This example extends the selection to the end of the current paragraph (including the paragraph mark) and then deletes the selection.

```
With Selection
    .EndOf Unit:=wdParagraph, Extend:=wdExtend
    .TypeBackspace
End With
```



TypeParagraph Method

-

Inserts a new, blank paragraph. If the selection isn't collapsed to an insertion point, it's replaced by the new paragraph. Use the **InsertParagraphAfter** or **InsertParagraphBefore** method to insert a new paragraph without deleting the contents of the selection.

Note This method corresponds to the functionality of the ENTER key.

expression.**TypeParagraph**

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

Example

This example collapses the selection to its end and then inserts a new paragraph following it.

```
With Selection  
    .Collapse Direction:=wdCollapseEnd  
    .TypeParagraph  
End With
```



TypeText Method

-

Inserts the specified text. If the [ReplaceSelection](#) property is **True**, the selection is replaced by the specified text. If **ReplaceSelection** is **False**, the specified text is inserted before the selection.

expression.**TypeText**(*Text*)

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

Text Required **String**. The text to be inserted.

Example

If **Typing replaces selection** is selected on the **Edit** tab in the **Options** dialog box, this example collapses the selection before inserting "Hello." This technique prevents existing document text from being replaced.

```
If Options.ReplaceSelection = True Then
    Selection.Collapse Direction:=wdCollapseStart
    Selection.TypeText Text:="Hello"
End If
```

This example inserts "Title" followed by a new paragraph.

```
Options.ReplaceSelection = False
With Selection
    .TypeText Text:="Title"
    .TypeParagraph
End With
```



Undo Method

Undoes the last action or a sequence of actions, which are displayed in the **Undo** list. Returns **True** if the actions were successfully undone.

expression.**Undo**(*Times*)

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

Times Optional **Variant**. The number of actions to be undone.

Example

This example undoes the last two actions taken in Sales.doc.

```
Documents("Sales.doc").Undo 2
```

This example undoes the last action. If the action is successfully undone, a message is displayed in the status bar.

```
On Error Resume Next  
If ActiveDocument.Undo = False Then _  
    StatusBar = "Undo was unsuccessful"
```



UndoClear Method

-

Clears the list of actions that can be undone for the specified document. Corresponds to the list of items that appears when you click the arrow beside the **Undo** button on the **Standard** toolbar.

Note 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").

expression.**UndoClear**

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

Example

This example clears the list of actions that can be undone for the active document.

```
ActiveDocument.UndoClear
```



Ungroup Method

-

Ungroups any grouped shapes in the specified shape or range of shapes. Disassembles pictures and OLE objects within the specified shape or range of shapes. Returns the ungrouped shapes as a single [ShapeRange](#) object.

expression.**Ungroup**

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

Remarks

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.

Example

This example ungroups any grouped shapes and disassembles any pictures or OLE objects on myDocument.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
    s.Ungroup
Next
```

This example ungroups any grouped shapes on myDocument without disassembling pictures or OLE objects on the document.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
    If s.Type = msoGroup Then s.Ungroup
Next
```



Unlink Method

-

Field object: Replaces the specified field with its most recent result.

Fields object: Replaces all the fields in the **Fields** collection with their most recent results.

expression.**Unlink**

expression Required. An expression that returns a **Field** or **Fields** object.

Remarks

When you unlink a field, its current result is converted to text or a graphic and can no longer be updated automatically. Note that some fields — such as XE (Index Entry) fields and SEQ (Sequence) fields — cannot be unlinked.

Example

This example unlinks the first field in "Sales.doc."

```
Documents("Sales.doc").Fields(1).Unlink
```

This example updates and unlinks all the fields in the first section in the active document.

```
With ActiveDocument.Sections(1).Range.Fields  
    .Update  
    .Unlink  
End With
```



Unload Method

Unloads all loaded add-ins and, depending on the value of the *RemoveFromList* argument, removes them from the **AddIns** collection.

expression.**Unload**(*RemoveFromList*)

expression Required. An expression that returns an **AddIns** object.

RemoveFromList Required **Boolean**. **True** to remove the unloaded add-ins from the **AddIns** collection (the names are removed from the **Templates and Add-ins** dialog box). **False** to leave the unloaded add-ins in the collection.

If the **Autoload** property for an unloaded add-in returns **True**, **Unload** cannot remove that add-in from the **AddIns** collection, regardless of the value of *RemoveFromList*.

Remarks

To unload a single template or WLL, set the [Installed](#) property of the **AddIn** object to **False**. To remove a single template or WLL from the **AddIns** collection, apply the [Delete](#) method to the **AddIn** object.

Example

This example unloads all the add-ins listed in the **Templates and Add-ins** dialog box. The add-in names remain in the **AddIns** collection.

```
If AddIns.Count > 0 Then AddIns.UnLoad RemoveFromList:=False
```



↳ [Show All](#)

Update Method

▶ [Update method as it applies to the **Field** object.](#)

Updates the result of the field object. When applied to a **Field** object, returns **True** if the field is updated successfully.

expression.**Update**

expression Required. An expression that returns one of the above objects.

▶ [Update method as it applies to the **Fields** object.](#)

Updates the result of the fields object. When applied to a **Fields** collection, returns 0 (zero) if no errors occur when the fields are updated, or returns a **Long** that represents the index of the first field that contains an error.

expression.**Update**

expression Required. An expression that returns one of the above objects.

▶ [Update method as it applies to the **Dialog**, **Index**, **LinkFormat**, **TableOfAuthorities**, **TableOfContents**, and **TableOfFigures** objects.](#)

Updates the values shown in a built-in Microsoft Word dialog box, updates the specified link, or updates the entries shown in specified index, table of authorities, table of figures or table of contents.

Note Use the [UpdatePageNumbers](#) method to update the page numbers of items in a table of contents or figures.

expression.**Update**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Fields** object.](#)

This example updates all the fields in the active document. A return value of 0 (zero) indicates that the fields were updated without error.

```
If ActiveDocument.Fields.Update = 0 Then
    MsgBox "Update Successful"
Else
    MsgBox "Field " & ActiveDocument.Fields.Update & _
        " has an error"
End If
```

This example updates any fields in the active document that aren't updated automatically.

```
For Each afield In ActiveDocument.Fields
    If afield.LinkFormat.AutoUpdate = False _
        Then afield.LinkFormat.Update
Next afield
```

▶ [As it applies to the **TableOfFigures** object.](#)

This example updates the first table of figures in the active document.

```
If ActiveDocument.TablesOfFigures.Count >= 1 Then
    ActiveDocument.TableOfFigures(1).Update
End If
```

▶ [As it applies to the **Field** object.](#)

This example updates the first field in the active document and displays a message in the status bar indicating whether or not the field was updated successfully.

```
If ActiveDocument.Fields(1).Update = True Then
    StatusBar = "Field updated"
Else
    StatusBar = "Error, field not updated"
End If
```

▶ [As it applies to the **Dialog** object.](#)

This example returns a **Dialog** object that refers to the Font dialog box. The font applied to the Selection object is changed to Arial, the dialog values are updated, and the Font dialog box is displayed.

```
Set myDialog = Dialogs(wdDialogFormatFont)
Selection.Font.Name = "Arial"
myDialog.Update
myDialog.Show
```



UpdateAutoFormat Method

-

Updates the table with the characteristics of a predefined table format. For example, if you apply a table format with **AutoFormat** and then insert rows and columns, the table may no longer match the predefined look.

UpdateAutoFormat restores the format.

expression.**UpdateAutoFormat**

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

Example

This example creates a table, applies a predefined format to it, adds a row, and then reapplies the predefined format.

```
Dim docNew As Document
Dim tableNew As Table

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 5, 5)

With tableNew
    .AutoFormat Format:=wdTableFormatColumns1
    .Rows.Add BeforeRow:=tableNew.Rows(1)
End With
MsgBox "Click OK to reapply autoformatting."
tableNew.UpdateAutoFormat
```

This example restores the predefined format to the table that contains the insertion point.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Tables(1).UpdateAutoFormat
Else
    MsgBox "The insertion point is not in a table."
End If
```



UpdateDocument Method

-
Updates the envelope in the document with the current envelope settings.

Note If you use this property before an envelope has been added to the document, an error occurs.

expression.**UpdateDocument**

expression Required. An expression that returns an **Envelope** object.

Example

This example formats the envelope in Report.doc to use a custom envelope size (4.5 inches by 7.5 inches).

```
Sub UpdateEnvelope()  
  
    On Error GoTo errhandler  
  
    With Documents("Report.doc").Envelope  
        .DefaultHeight = InchesToPoints(4.5)  
        .DefaultWidth = InchesToPoints(7.5)  
        .UpdateDocument  
    End With  
  
    Exit Sub  
  
errhandler:  
  
    If Err = 5852 Then _  
        MsgBox "Report.doc doesn't include an envelope"  
  
End Sub
```

This example adds an envelope to the active document, using predefined addresses. The default envelope bar code and Facing Identification Mark (FIM-A) settings are set to **True**, and the envelope in the active document is updated.

```
Dim strAddress As String  
Dim strReturn As String  
  
strAddress = "Darlene Rudd" & vbCr & "1234 E. Main St." _  
    & vbCr & "Our Town, WA 98004"  
strReturn = "Patricia Reed" & vbCr & "N. 33rd St." _  
    & vbCr & "Other Town, WA 98040"  
ActiveDocument.Envelope.Insert _  
    Address:=strAddress, ReturnAddress:=strReturn  
With ActiveDocument.Envelope  
    .DefaultPrintBarCode = True  
    .DefaultPrintFIMA = True  
    .UpdateDocument  
End With
```



UpdatePageNumbers Method

-
Updates the page numbers for items in the specified table of contents or table of figures.

expression.**UpdatePageNumbers**

expression Required. An expression that returns a **TableOfContents** or **TableOfFigures** object.

Example

This example updates all tables of figures in Sales.doc.

```
Dim tofLoop As TableOfFigures
```

```
For Each tofLoop In Documents("Sales.doc").TablesOfFigures  
    tofLoop.UpdatePageNumbers
```

```
Next tofLoop
```

This example inserts a page break at the insertion point and then updates the page numbers for the first table of contents in the active document.

```
Selection.Collapse Direction:=wdCollapseStart
```

```
Selection.InsertBreak Type:=wdPageBreak
```

```
ActiveDocument.TablesOfContents(1).UpdatePageNumbers
```



UpdateSource Method

-

Saves the changes made to the results of an INCLUDETEXT field back to the source document.

Note The source document must be formatted as a Word document.

expression.**UpdateSource**

expression Required. An expression that returns a **Field** or **Fields** object.

Example

This example updates the INCLUDETEXT fields in the active document.

```
Dim fldLoop As Field
```

```
For Each fldLoop In ActiveDocument.Fields  
    If fldLoop.Type = wdFieldIncludeText Then _  
        fldLoop.UpdateSource  
Next fldLoop
```



UpdateStyles Method

-
Copies all styles from the attached template into the document, overwriting any existing styles in the document that have the same name.

expression.**UpdateStyles**

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

Example

This example copies the styles from the attached template into each open document, and then it closes each document.

```
For Each aDoc In Documents
    aDoc.UpdateStyles
    aDoc.Close SaveChanges:=wdSaveChanges
Next aDoc
```

This example changes the formatting of the Heading 1 style in the template attached to the active document. The **UpdateStyles** method updates the styles in the active document, including the Heading 1 style.

```
Set aDoc = ActiveDocument.AttachedTemplate.OpenAsDocument
With aDoc.Styles(wdStyleHeading1).Font
    .Name = "Arial"
    .Bold = False
End With
aDoc.Close SaveChanges:=wdSaveChanges
ActiveDocument.UpdateStyles
```



UpdateSummaryProperties Method

-
Updates the keyword and comment text in the **Properties** dialog box (**File** menu) to reflect the AutoSummary content for the specified document.

expression.**UpdateSummaryProperties**

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

Example

This example highlights key points in the active document and updates the summary information in the **Properties** dialog box (**File** menu).

```
With ActiveDocument
    .AutoSummarize Length:=wd25Percent, _
        Mode:=wdSummaryModeHighlight
    .UpdateSummaryProperties
End With
```



UseDefaultFolderSuffix Method

-

Sets the folder suffix for the specified document to the default suffix for the language support you have selected or installed.

expression.**UseDefaultFolderSuffix**

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

Remarks

Microsoft Word uses the folder suffix when you save a document as a Web page, use long file names, and choose to save supporting files in a separate folder (that is, if the [UseLongFileNames](#) and [OrganizeInFolder](#) properties are set to **True**).

The suffix appears in the folder name after the document name. For example, if the document is called "Doc1" and the language is English, the folder name is Doc1_files. The available folder suffixes are listed in the [FolderSuffix](#) property topic.

Example

This example sets the folder suffix for the active document to the default suffix.

```
ActiveDocument.WebOptions.UseDefaultFolderSuffix
```



UserPicture Method

-

Fills the specified shape with one large image. If you want to fill the shape with small tiles of an image, use the [UserTextured](#) method.

expression.**UserPicture**(*PictureFile*)

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

PictureFile Required **String**. The name of the picture file.

Example

This example adds two rectangles to the active document. The rectangle on the left is filled with one large image of the picture in Tiles.bmp; the rectangle on the right is filled with many small tiles of the picture in Tiles.bmp.

```
Sub Pic()  
    ' Windows NT and Windows2000 users need to  
    ' specify a different explicit path to a bitmap  
    ' file in the methods below.  
    With ActiveDocument.Shapes  
        .AddShape(msoShapeRectangle, 0, 0, 200, 100).Fill _  
            .UserPicture "C:\Windows\Tiles.bmp"  
        .AddShape(msoShapeRectangle, 300, 0, 200, 100).Fill _  
            .UserTextured "C:\Windows\Tiles.bmp"  
    End With  
End Sub
```



UserTextured Method

-

Fills the specified shape with small tiles of an image. If you want to fill the shape with one large image, use the **UserPicture** method.

expression.**UserTextured**(*TextureFile*)

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

TextureFile Required **String**. The name of the picture file.

Example

This example adds two rectangles to the active document. The rectangle on the left is filled with one large image of the picture in Tiles.bmp; the rectangle on the right is filled with many small tiles of the picture in Tiles.bmp

```
Sub Texture()  
    ' Windows NT and Windows2000 users need to  
    ' specify a different explicit path to a bitmap  
    ' file in the methods below.  
    With ActiveDocument.Shapes  
        .AddShape(msoShapeRectangle, 0, 0, 200, 100).Fill _  
            .UserPicture "C:\Windows\Tiles.bmp"  
        .AddShape(msoShapeRectangle, 300, 0, 200, 100).Fill _  
            .UserTextured "C:\Windows\Tiles.bmp"  
    End With  
End Sub
```



ValidLinkTarget Method

-

Determines whether the text frame of one shape can be linked to the text frame of another shape. Returns **True** if *TargetTextFrame* is a valid target. Returns **False** if *TargetTextFrame* already contains text or is already linked, or if the shape doesn't support attached text.

expression.ValidLinkTarget(*TargetTextFrame*)

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

TargetTextFrame Required **TextFrame** object. The target text frame that you'd like to link the text frame returned by *expression* to.

Example

This example checks to see whether the text frames for the first and second shapes in the active document can be linked to one another. If so, the example links the two text frames.

```
Dim textFrame1 As TextFrame
Dim textFrame2 As TextFrame

Set textFrame1 = ActiveDocument.Shapes(1).TextFrame
Set textFrame2 = ActiveDocument.Shapes(2).TextFrame
If textFrame1.ValidLinkTarget(textFrame2) = True Then
    textFrame1.Next = textFrame2
End If
```



ViewCode Method

-

Displays the code window for the selected ActiveX control in the specified document.

Note This method is available only from outside of Word.

expression.**ViewCode**

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

ViewPropertyBrowser Method

-

Displays the property window for the selected ActiveX control in the specified document.

Note This method is available only from outside of Word.

expression.**ViewPropertyBrowser**

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

WebPagePreview Method

-
Displays a preview of the current document as it would look if saved as a Web page.

expression.**WebPagePreview**

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

Example

This example displays the current document as it would appear if saved as a Web page.

ActiveDocument.**WebPagePreview**



↳ [Show All](#)

WholeStory Method

-
Expands a range or selection to include the entire [story](#).

expression.**WholeStory**

expression Required. An expression that returns a **Range** or **Selection** object.

Remarks

The following instructions, where myRange is a valid **Range** object, are functionally equivalent:

```
myRange.WholeStory  
myRange.Expand Unit:=wdStory
```

Example

This example expands myRange to include the entire story and then applies the Arial font to the range.

```
Set myRange = Selection.Range
myRange.WholeStory
myRange.Font.Name = "Arial"
```

This example expands myRange to include the entire comments story (**wdCommentsStory**) and then copies the comments into a new document.

```
If ActiveDocument.Comments.Count >= 1 Then
    Set myRange = Activatedocument.Comments(1).Range
    myRange.WholeStory
    myRange.Copy
    Documents.Add.Content.Paste
End If
```



↳ [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 document and then places the oval as second from the back in the z-order if there is at least one other shape on the document.

```
With ActiveDocument.Shapes.AddShape(Type:=msoShapeOval, Left:=100, _  
    Top:=100, Width:=100, Height:=300)  
    While .ZOrderPosition > 2  
        .ZOrder msoSendBackward  
    Wend  
End With
```



Accent Property

-
True if a vertical accent bar separates the callout text from the callout line.
Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

Example

This example adds an oval to the active document 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.

```
Dim shapeCallout As Shape
```

```
With ActiveDocument.Shapes
```

```
    .AddShape msoShapeOval, 180, 200, 280, 130
```

```
    Set shapeCallout = .AddCallout(msoCalloutTwo, 420, 170, 170, 40)
```

```
    With shapeCallout
```

```
        .TextFrame.TextRange.Text = "My oval"
```

```
        .Callout.Accent = msoTrue
```

```
        .Callout.Border = msoFalse
```

```
    End With
```

```
End With
```



AccentedLetters Property

-
True if the specified index contains separate headings for accented letters (for example, words that begin with "À" are under one heading and words that begin with "A" are under another). Read/write **Boolean**.

Example

This example formats the first index in the active document in a single column, with the appropriate letter preceding each alphabetic group and separate headings for accented letters.

```
If ActiveDocument.Indexes.Count >= 1 Then
  With ActiveDocument.Indexes(1)
    .HeadingSeparator = wdHeadingSeparatorLetter
    .NumberOfColumns = 1
    .AccentedLetters = True
  End With
End If
```



↳ [Show All](#)

Active Property

▶ [Active property as it applies to the **LineNumbering** object.](#)

True if line numbering is active for the specified document, section, or sections. Read/write **Long**.

expression.Active

expression Required. An expression that returns a [LineNumbering](#) object.

▶ [Active property as it applies to the **Selection** object.](#)

True if the selection in the specified window or pane is active. Read-only **Boolean**.

expression.Active

expression Required. An expression that returns a [Selection](#) object.

▶ [Active property as it applies to the **Window** object.](#)

True if the specified window is active. Read-only **Boolean**.

expression.Active

expression Required. An expression that returns a [Window](#) object.

Example

▶ [As it applies to the **LineNumbering** object.](#)

This example activates line numbering for the first section in the selection.

```
Sub CountByFive()  
    With Selection.Sections(1).PageSetup.LineNumbering  
        .Active = True  
        .CountBy = 5  
        .StartingNumber = 1  
    End With  
End Sub
```

▶ [As it applies to the **Selection** object.](#)

This example splits the active window into two panes and activates the selection in the first pane, if it isn't already active.

```
Sub SplitWindow()  
    ActiveDocument.ActiveWindow.Split = True  
    If ActiveDocument.ActiveWindow.Panes(1).Selection _  
        .Active = False Then  
        ActiveDocument.ActiveWindow.Panes(1).Activate  
    End If  
End Sub
```

▶ [As it applies to the **Window** object.](#)

This example activates the first window in the **Windows** collection, if the window isn't currently active.

```
Sub ActiveWin()  
    If Windows(1).Active = False Then Windows(1).Activate  
End Sub
```



ActiveCustomDictionary Property

-
Returns or sets a [Dictionary](#) object that represents the custom dictionary to which words will be added. Read/write.

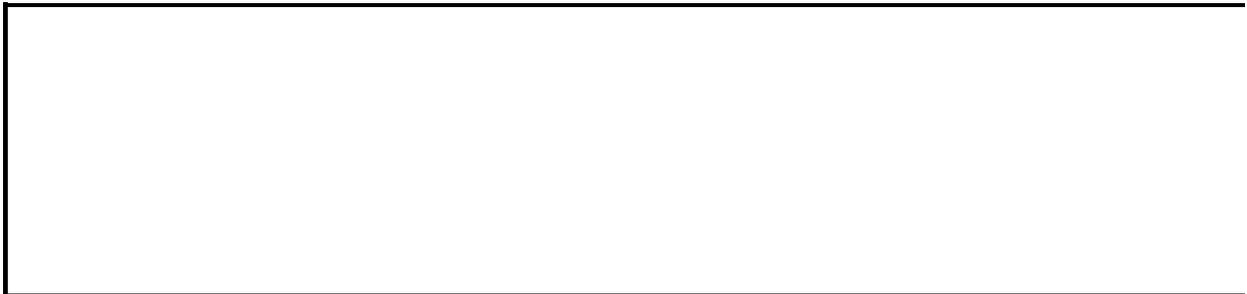
Example

This example displays the full path and file name of the active custom dictionary.

```
Set dicCustom = Application.CustomDictionaries.ActiveCustomDictionary  
MsgBox dicCustom.Path & Application.PathSeparator & dicCustom.Name
```

This example clears all existing custom dictionaries, adds a custom dictionary named "Home.dic," and then loads the new dictionary.

```
Dim dicCustom As Dictionary  
  
Application.CustomDictionaries.ClearAll  
  
Set dicCustom = Application.CustomDictionaries _  
    .Add(FileName:="C:\Program Files" _  
        & "\Microsoft Office\Office\Home.dic")  
Application.CustomDictionaries.ActiveCustomDictionary = dicCustom
```



ActiveDocument Property

-

Returns a [Document](#) object that represents the active document (the document with the focus). If there are no documents open, an error occurs. Read-only.

Example

This example displays the name of the active document, or if there are no documents open, it displays a message.

```
If Application.Documents.Count >= 1 Then
    MsgBox ActiveDocument.Name
Else
    MsgBox "No documents are open"
End If
```

This example collapses the selection to an insertion point and then creates a range for the next five characters in the selection.

```
Dim rngTemp As Range

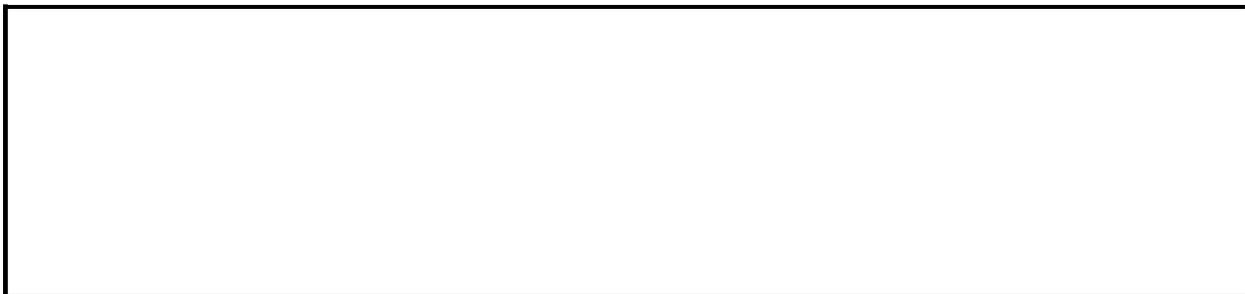
Selection.Collapse Direction:=wdCollapseStart
Set rngTemp = ActiveDocument.Range(Start:=Selection.Start, _
    End:=Selection.Start + 5)
```

This example inserts texts at the beginning of the active document and then prints the document.

```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
With rngTemp
    .InsertBefore "Company Report"
    .Font.Name = "Arial"
    .Font.Size = 24
    .InsertParagraphAfter
End With

ActiveDocument.PrintOut
```



ActiveGrammarDictionary Property

-

Returns a [Dictionary](#) object that represents the active grammar dictionary for the specified language. Read-only.

Remarks

If there's no grammar dictionary installed for the specified language, this property returns **Nothing**.

Example

This example displays the full path and file name of the active grammar dictionary.

```
Dim lngLanguage As Long
Dim dicGrammar As Dictionary

lngLanguage = Selection.LanguageID
Set dicGrammar = Languages(lngLanguage).ActiveGrammarDictionary
MsgBox dicGrammar.Path & Application.PathSeparator & dicGrammar.Name
```



ActiveHyphenationDictionary Property

Returns a [Dictionary](#) object that represents the active hyphenation dictionary for the specified language. Read-only.

Remarks

If there's no hyphenation dictionary installed for the specified language, this property returns **Nothing**.

Example

This example displays the full path and file name of the active hyphenation dictionary.

```
Dim lngLanguage As Long
Dim dicHyphen As Dictionary

lngLanguage = Selection.LanguageID
Set dicHyphen = Languages(lngLanguage).ActiveHyphenationDictionary
If dicHyphen Is Nothing Then
    MsgBox "No hyphenation dictionary installed!"
Else
    MsgBox dicHyphen.Path & Application.PathSeparator & dicHyphen.Name
End If
```



ActivePane Property

-

Returns a [Pane](#) object that represents the active pane for the specified window.
Read-only.

Example

This example splits the active window and then activates the next pane after the active pane.

```
With ActiveDocument.ActiveWindow
    .Split = True
    .ActivePane.Next.Activate
    MsgBox "Pane " & .ActivePane.Index & " is active"
End With
```

This example activates the first window and displays tabs in the active pane.

```
With Application.Windows(1)
    .Activate
    .ActivePane.View.ShowTabs = True
End With
```



ActivePrinter Property

-
Returns or sets the name of the active printer. Read/write **String**.

Example

This example displays the name of the active printer.

```
MsgBox "The name of the active printer is " & ActivePrinter
```

This example makes a network HP LaserJet IIISi printer the active printer.

```
Application.ActivePrinter = "HP LaserJet IIISi on \\printers\laser"
```

This example makes a local HP LaserJet 4 printer on LPT1 the active printer.

```
Application.ActivePrinter = "HP LaserJet 4 local on LPT1:"
```

ActiveRecord Property

-

Returns or sets the active mail merge data record. Can be either a valid data record number in the query result or one of the following read/write [WdMailMergeActiveRecord](#) constants.

WdMailMergeActiveRecord can be one of these WdMailMergeActiveRecord constants.

wdLastRecord

wdNoActiveRecord

wdFirstRecord

wdNextRecord

wdPreviousRecord

Note 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.

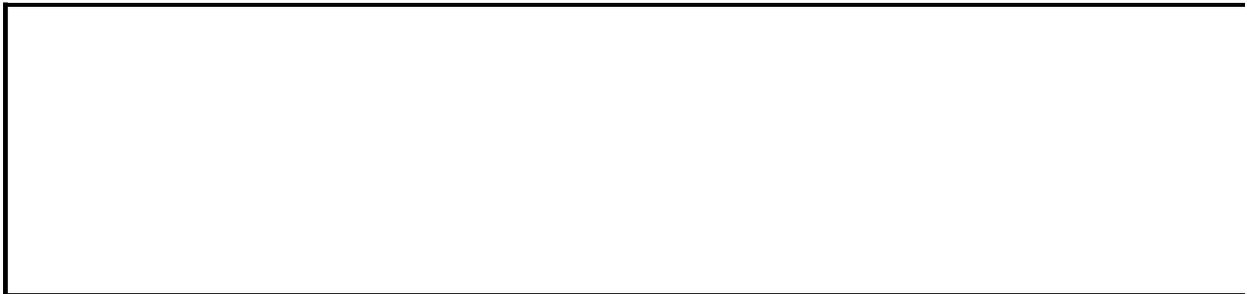
Example

This example hides the mail merge field codes in the active document so that the merge data is visible in the main document. The active record is then advanced to the next record in the data source.

```
If ActiveDocument.MailMerge.MainDocumentType <> _  
    wdNotAMergeDocument Then  
    With ActiveDocument.MailMerge  
        .ViewMailMergeFieldCodes = False  
        .DataSource.ActiveRecord = wdNextRecord  
    End With  
End If
```

This example returns the numerical position of the active data record from Main2.doc.

```
Dim intRecordNumber as Integer  
  
If Documents("Main2.doc").MailMerge.State = _  
    wdMainAndDataSource Or _  
    wdMainAndSourceAndHeader Then  
    intRecordNumber = Documents("Main2.doc").MailMerge _  
        .DataSource.ActiveRecord  
End If
```



ActiveSpellingDictionary Property

Returns a [Dictionary](#) object that represents the active spelling dictionary for the specified language.

expression.ActiveSpellingDictionary

expression Required. An expression that returns a [Language](#) object.

Remarks

If there's no spelling dictionary installed for the specified language, this property returns **Nothing**.

Example

This example returns the full path and file name of the active spelling dictionary.

```
Dim lngLanguage As Long
Dim dicSpelling As Dictionary

lngLanguage = Selection.LanguageID
Set dicSpelling = Languages(lngLanguage).ActiveSpellingDictionary
If dicSpelling Is Nothing Then
    MsgBox "No spelling dictionary installed!"
Else
    MsgBox dicSpelling.Path & Application.PathSeparator _
        & dicSpelling.Name
End If
```



↳ [Show All](#)

ActiveTheme Property

-

Returns the name of the active [theme](#) plus the theme formatting options for the specified document. Returns "none" if the document doesn't have an active theme. Read-only **String**.

Remarks

For an explanation of the value returned by this property, see the *Name* argument of the [ApplyTheme](#) method. The value returned by this property may not correspond to the theme's display name. To return a theme's display name, use the [ActiveThemeDisplayName](#) property.

Example

This example applies a theme and then displays the name of the active theme plus the theme formatting options for the current document.

```
Sub CheckTheme()  
    ActiveDocument.ApplyTheme "artsy 100"  
    MsgBox ActiveDocument.ActiveTheme  
End Sub
```



↳ [Show All](#)

ActiveThemeDisplayName Property

-

Returns the display name of the active [theme](#) for the specified document.
Returns "none" if the document doesn't have an active theme. Read-only **String**.

Remarks

A theme's display name is the name that appears in the **Theme** dialog box. This name may not correspond to the string you would use to set a default theme or to apply a theme to a document.

Example

This example returns the display name of the active theme for the current document.

```
Sub DisplayThemeName()  
    ActiveDocument.ApplyTheme "artsy 100"  
    MsgBox ActiveDocument.ActiveThemeDisplayName  
End Sub
```



ActiveThesaurusDictionary Property

Returns a [Dictionary](#) object that represents the active thesaurus dictionary for the specified language.

expression.ActiveThesaurusDictionary

expression Required. An expression that returns a [Language](#) object.

Remarks

If there's no thesaurus dictionary installed for the specified language, this property returns **Nothing**.

Example

This example returns the full path and file name of the active thesaurus dictionary.

```
Dim lngLanguage As Long
Dim dicThesaurus As Dictionary

lngLanguage = Selection.LanguageID
Set dicThesaurus = Languages(lngLanguage).ActiveThesaurusDictionary
If dicThesaurus Is Nothing Then
    MsgBox "No thesaurus dictionary installed!"
Else
    MsgBox dicThesaurus.Path & Application.PathSeparator _
        & dicThesaurus.Name
End If
```



ActiveWindow Property

-

Returns a [Window](#) object that represents the active window (the window with the focus). If there are no windows open, an error occurs. Read-only.

Example

This example displays the caption text for the active window.

```
Sub WindowCaption()  
    MsgBox ActiveDocument.ActiveWindow.Caption  
End Sub
```

This example opens a new window for the active window of the active document and then tiles all the windows.

```
Sub WindowTiled()  
    Dim wndTileWindow As Window  
  
    Set wndTileWindow = ActiveDocument.ActiveWindow.NewWindow  
    Windows.Arrange ArrangeStyle:=wdTiled  
End Sub
```

This example splits the first document window.

```
Sub WindowSplit()  
    Documents(1).ActiveWindow.Split = True  
End Sub
```



↳ [Show All](#)

ActiveWritingStyle Property

Returns or sets the writing style for a specified language in the specified document. Read/write **String**.

Note The [WritingStyleList](#) property returns an array of the names of the available writing styles.

expression.ActiveWritingStyle(LanguageID)

expression Required. An expression that returns a [Document](#) object.

LanguageID Required **Variant**. The language to set the writing style for in the specified document. Can be either a string or one of the following [WdLanguageID](#) constants.

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt

wdArabicIraq

wdArabicJordan

wdArabicKuwait

wdArabicLebanon

wdArabicLibya

wdArabicMorocco

wdArabicOman

wdArabicQatar

wdArabicSyria

wdArabicTunisia
wdArabicUAE
wdArabicYemen
wdArmenian
wdAssamese
wdAzeriCyrillic
wdAzeriLatin
wdBasque
wdBelgianDutch
wdBelgianFrench
wdBengali
wdBosniaHerzegovina
wdBrazilianPortuguese
wdBulgarian
wdBurmese
wdByelorussian
wdCatalan
wdChineseHongKong
wdChineseMacao
wdChineseSingapore
wdCroatian
wdCzech
wdDanish
wdDutch
wdEnglishAUS
wdEnglishBelize
wdEnglishCanadian
wdEnglishCaribbean
wdEnglishIreland
wdEnglishJamaica
wdEnglishNewZealand
wdEnglishPhilippines
wdEnglishSouthAfrica

wdEnglishTrinidad
wdEnglishUK
wdEnglishUS
wdEnglishZimbabwe
wdEstonian
wdFaeroese
wdFarsi
wdFinnish
wdFrench
wdFrenchCameroon
wdFrenchCanadian
wdFrenchCotedIvoire
wdFrenchLuxembourg
wdFrenchMali
wdFrenchMonaco
wdFrenchReunion
wdFrenchSenegal
wdFrenchWestIndies
wdFrenchZaire
wdFrisianNetherlands
wdGaelicIreland
wdGaelicScotland
wdGalician
wdGeorgian
wdGerman
wdGermanAustria
wdGermanLiechtenstein
wdGermanLuxembourg
wdGreek
wdGujarati
wdHebrew
wdHindi
wdHungarian

wdIcelandic
wdIndonesian
wdItalian
wdJapanese
wdKannada
wdKashmiri
wdKazakh
wdKhmer
wdKirghiz
wdKonkani
wdKorean
wdLanguageNone
wdLao
wdLatvian
wdLithuanian
wdLithuanianClassic
wdMacedonian
wdMalayalam
wdMalayBruneiDarussalam
wdMalaysian
wdMaltese
wdManipuri
wdMarathi
wdMexicanSpanish
wdMongolian
wdNepali
wdNoProofing
wdNorwegianBokmol
wdNorwegianNynorsk
wdOriya
wdPolish
wdPortuguese
wdPunjabi

wdRhaetoRomanic
wdRomanian
wdRomanianMoldova
wdRussian
wdRussianMoldova
wdSamiLappish
wdSanskrit
wdSerbianCyrillic
wdSerbianLatin
wdSesotho
wdSimplifiedChinese
wdSindhi
wdSlovak
wdSlovenian
wdSorbian
wdSpanish
wdSpanishArgentina
wdSpanishBolivia
wdSpanishChile
wdSpanishColombia
wdSpanishCostaRica
wdSpanishDominicanRepublic
wdSpanishEcuador
wdSpanishElSalvador
wdSpanishGuatemala
wdSpanishHonduras
wdSpanishModernSort
wdSpanishNicaragua
wdSpanishPanama
wdSpanishParaguay
wdSpanishPeru
wdSpanishPuertoRico
wdSpanishUruguay

wdSpanishVenezuela
wdSutu
wdSwahili
wdSwedish
wdSwedishFinland
wdSwissFrench
wdSwissGerman
wdSwissItalian
wdTajik
wdTamil
wdTatar
wdTelugu
wdThai
wdTibetan
wdTraditionalChinese
wdTsonga
wdTswana
wdTurkish
wdTurkmen
wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin
wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdZulu

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 sets the writing style used for French, German, and U.S. English for the active document. You must have the grammar files installed for French, German, and U.S. English to run this example.

```
With ActiveDocument
    .ActiveWritingStyle(wdFrench) = "Commercial"
    .ActiveWritingStyle(wdGerman) = "Technisch/Wiss"
    .ActiveWritingStyle(wdEnglishUS) = "Technical"
End With
```

This example returns the writing style for the language of the selection.

```
Sub WhichLanguage()
    Dim varLang As Variant

    varLang = Selection.LanguageID
    MsgBox ActiveDocument.ActiveWritingStyle(varLang)
End Sub
```



AddBiDirectionalMarksWhenSavingT Property

True if Microsoft Word adds bidirectional control characters when saving a document as a text file. Read/write **Boolean**.

expression.**AddBiDirectionalMarksWhenSavingTextFile**

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

Remarks

Saving text files with bidirectional control characters preserves right-to-left and left-to-right properties and the order of neutral characters.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Word to add bidirectional control characters when saving a document as a text file.

```
Options.AddBiDirectionalMarksWhenSavingTextFile = True
```



AddControlCharacters Property

-
True if Microsoft Word adds bidirectional control characters when cutting and copying text. Read/write **Boolean**.

expression.**AddControlCharacters**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Word to add bidirectional control characters when cutting and copying text.

```
Options.AddControlCharacters = True
```



AddHebDoubleQuote Property

True if Microsoft Word encloses number formats in double quotation marks (").
Read/write **Boolean**.

expression.**AddHebDoubleQuote**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Word to enclose number formats in double quotation marks ("").

```
Options.AddHebDoubleQuote = True
```



AddIns Property

Returns an [AddIns](#) collection that represents all available add-ins, regardless of whether they're currently loaded. The **AddIns** collection includes the global templates and Word add-in libraries (WLLs) listed in the **Templates and Add-ins** dialog box (**Tools** menu). Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example returns the total number of add-ins.

```
Dim intAddIns as Integer  
intAddIns = AddIns.Count
```

This example displays the name of each add-in in the **Addins** collection.

```
Dim addinLoop as AddIn  
For Each addinLoop In AddIns  
    MsgBox addinLoop.Name  
Next addinLoop
```



↳ [Show All](#)

Address Property

▶ [Address property as it applies to the **Envelope** object.](#)

Returns the envelope delivery address as a **Range** object. Read-only.

expression.**Address**

expression Required. An expression that returns one of the above objects.

Note An error occurs if you use this property when there hasn't been an envelope added to the specified document.

▶ [Address property as it applies to the **Hyperlink** object.](#)

Returns or sets the address (for example, a file name or URL) of the specified hyperlink. Read/write **String**.

expression.**Address**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Envelope** object.](#)

This example displays the delivery address if an envelope has been added to the document; otherwise, it displays a message.

```
On Error GoTo errhandler
addr = ActiveDocument.Envelope.Address.Text
MsgBox Prompt:=addr, Title:="Delivery Address"
errhandler:
If Err = 5852 Then MsgBox "Insert an envelope into the document"
```

▶ [As it applies to the **Hyperlink** object.](#)

This example adds a hyperlink to the selection in the active document, sets the address, and then displays the address in a message box.

```
Set aHLink = ActiveDocument.Hyperlinks.Add( _
    Anchor:=Selection.Range, _
    Address:="http://forms")
MsgBox "The hyperlink goes to " & aHLink.Address
```

If the active document includes hyperlinks, this example inserts a list of the hyperlink destinations at the end of the document.

```
Set myRange = ActiveDocument _
    .Range(Start:=ActiveDocument.Content.End - 1)
Count = 0
For Each aHyperlink In ActiveDocument.Hyperlinks
    Count = Count + 1
    With myRange
        .InsertAfter "Hyperlink #" & Count & vbCrLf
        .InsertAfter aHyperlink.Address
        .InsertParagraphAfter
    End With
Next aHyperlink
```



AddressFromLeft Property

Returns or sets the distance (in points) between the left edge of the envelope and the delivery address. Read/write **Single**.

Note If you use this property before an envelope has been added to the document, an error occurs.

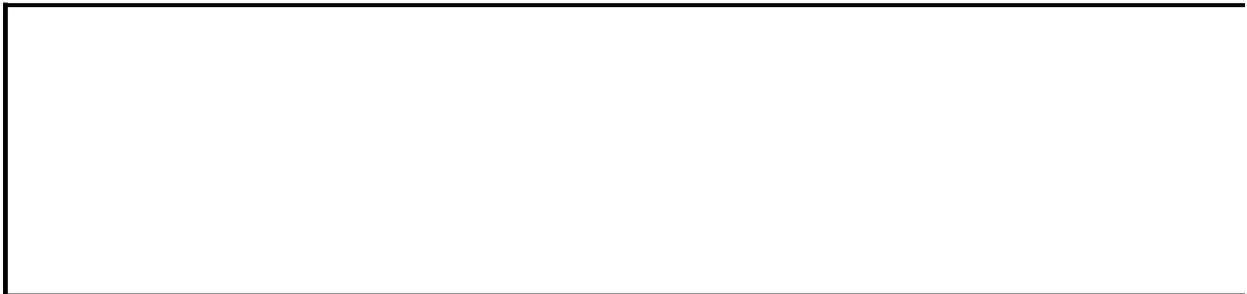
Example

This example creates a new document and adds an envelope with a predefined delivery address and return address. The example then sets the distance between the left edge of the envelope and the delivery address to 3.75 inches.

```
Dim strAddress As String
Dim strReturn As String

strAddress = "James Allard" & vbCrLf & "123 Skye St." & vbCrLf _
    & "Our Town, WA 98004"
strReturn = "Rich Andrews" & vbCrLf & "123 Main" & vbCrLf _
    & "Other Town, WA 98004"

With Documents.Add.Envelope
    .Insert Address:=strAddress, ReturnAddress:=strReturn
    .AddressFromLeft = InchesToPoints(3.75)
End With
ActiveDocument.ActiveWindow.View.Type = wdPrintView
```



AddressFromTop Property

-

Returns or sets the distance (in points) between the top edge of the envelope and the delivery address. Read/write **Single**.

Note If you use this property before an envelope has been added to the document, an error occurs.

Example

This example creates a new document and adds an envelope with a predefined delivery address and return address. The example then sets the distance between the top edge of the envelope and the delivery address to 1.75 inches and sets the distance between the left edge of the envelope and the delivery address is set to 3.75 inches.

```
Dim strAddress As String
Dim strReturn As String

strAddress = "Michael Bunney" & vbCrLf & "123 Skye St." & vbCrLf _
    & "Our Town, WA 98040"
strReturn = "Kate Dresen" & vbCrLf & "123 Main" & vbCrLf _
    & "Other Town, WA 98040"

With Documents.Add.Envelope
    .Insert Address:=strAddress, ReturnAddress:=strReturn
    .AddressFromTop = InchesToPoints(1.75)
    .AddressFromLeft = InchesToPoints(3.75)
End With

ActiveDocument.ActiveWindow.View.Type = wdPrintView
```



AddressStyle Property

-
Returns a [Style](#) object that represents the delivery address style for the envelope.
Read-only

Note If an envelope is added to the document, text formatted with the Envelope Address style is automatically updated.

Example

This example modifies the font formatting associated with the Envelope Address style.

```
With ActiveDocument.Envelope.AddressStyle.Font
    .Bold = False
    .Name = "Times New Roman"
    .Size = 16
End With
```



AddSpaceBetweenFarEastAndAlpha Property

True if Microsoft Word is set to automatically add spaces between Japanese and Latin text for the specified paragraphs. This property returns **wdUndefined** if it's set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example sets Microsoft Word to automatically add spaces between Japanese and Latin text for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).AddSpaceBetweenFarEastAndAlpha = True
```



AddSpaceBetweenFarEastAndDigit Property

True if Microsoft Word is set to automatically add spaces between Japanese text and numbers for the specified paragraphs. This property returns **wdUndefined** if it's set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example sets Microsoft Word to automatically add spaces between Japanese text and numbers for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).AddSpaceBetweenFarEastAndDigit = True
```



↳ [Show All](#)

Adjustments Property

▶ [Adjustments property as it applies to the **Shape** object.](#)

Returns an **Adjustments** object that contains adjustment values for all the adjustments in the specified **Shape** object that represents an AutoShape or WordArt. Read-only.

▶ [Adjustments property as it applies to the **ShapeRange** object.](#)

Returns an **Adjustments** object that contains adjustment values for all the adjustments in the specified **ShapeRange** object that represents an AutoShape or WordArt. Read-only.

Example

▶ [As it applies to the **Shape** object.](#)

This example sets to 0.25 the value of adjustment one on shape three on myDocument.

```
Set myDocument = ActiveDocument  
myDocument.Shapes(3).Adjustments(1) = 0.25
```



↳ [Show All](#)

Alignment Property

▶ [Alignment property as it applies to the **HorizontalLineFormat** object.](#)

Returns or sets a **WdHorizontalLineAlignment** constant that represents the alignment for the specified horizontal line. Read/write.

WdHorizontalLineAlignment can be one of these WdHorizontalLineAlignment constants.

wdHorizontalLineAlignCenter

wdHorizontalLineAlignRight

wdHorizontalLineAlignLeft

expression.Alignment

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

▶ [Alignment property as it applies to the **ListLevel** object.](#)

Returns or sets a **WdListLevelAlignment** constant that represents the alignment for the list level of the list template. Read/write.

WdListLevelAlignment can be one of these WdListLevelAlignment constants.

wdListLevelAlignLeft

wdListLevelAlignCenter

wdListLevelAlignRight

expression.Alignment

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

▶ [Alignment property as it applies to the **PageNumber** object.](#)

Returns or sets a [WdPageNumberAlignment](#) constant that represents the alignment for the page number. Read/write.

WdPageNumberAlignment can be one of these WdPageNumberAlignment constants.

wdAlignPageNumberInside
wdAlignPageNumberOutside
wdAlignPageNumberCenter
wdAlignPageNumberLeft
wdAlignPageNumberRight

expression.Alignment

expression Required. An expression that returns a [PageNumber](#) object.

► [Alignment property as it applies to the Paragraph, ParagraphFormat, and Paragraphs objects.](#)

Returns or sets a [WdParagraphAlignment](#) constant that represents the alignment for the specified paragraphs. Read/write.

WdParagraphAlignment can be one of these WdParagraphAlignment constants.

wdAlignParagraphCenter
wdAlignParagraphDistribute
wdAlignParagraphJustify
wdAlignParagraphJustifyHi
wdAlignParagraphJustifyLow
wdAlignParagraphJustifyMed
wdAlignParagraphLeft
wdAlignParagraphRight
wdAlignParagraphThaiJustify

expression.Alignment

expression Required. An expression that returns a [Paragraph](#), [ParagraphFormat](#), or [Paragraphs](#) object.

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.

▶ [Alignment property as it applies to the **Row**, **Rows**, and **TableStyle** objects.](#)

Returns or sets a **WdRowAlignment** constant that represents the alignment for the specified rows. Read/write.

WdRowAlignment can be one of these WdRowAlignment constants.

wdAlignRowLeft

wdAlignRowCenter

wdAlignRowRight

expression.**Alignment**

expression Required. An expression that returns a **Row**, **Rows**, or **TableStyle** object.

▶ [Alignment property as it applies to the **TabStop** object.](#)

Returns or sets a **WdTabAlignment** constant that represents the alignment for the specified tab stop. Read/write.

WdTabAlignment can be one of these WdTabAlignment constants.

wdAlignTabBar

wdAlignTabCenter

wdAlignTabDecimal

wdAlignTabLeft

wdAlignTabList

wdAlignTabRight

expression.**Alignment**

expression Required. An expression that returns a [TabStop](#) object.

► [Alignment property as it applies to the TextEffectFormat object.](#)

Returns or sets an [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.

Example

▶ [As it applies to the **Paragraph** object.](#)

This example right-aligns the first paragraph in the active document.

```
Sub AlignParagraph()  
    ActiveDocument.Paragraphs(1).Alignment = _  
        wdAlignParagraphRight  
End Sub
```

▶ [As it applies to the **Rows** object.](#)

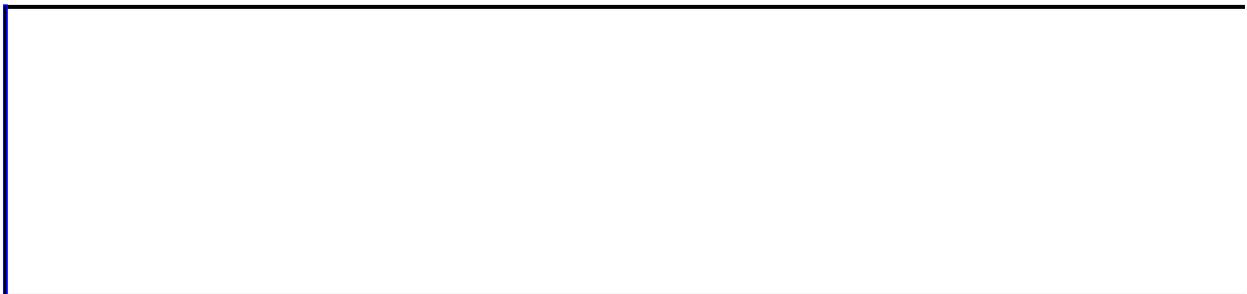
This example centers all the rows in the first table of the active document.

```
Sub CenterRows()  
    ActiveDocument.Tables(1).Rows _  
        .Alignment = wdAlignRowCenter  
End Sub
```

▶ [As it applies to the **TabStop** object.](#)

This example centers the first tab stop in the first paragraph of the active document.

```
Sub CenterTabStop()  
    ActiveDocument.Paragraphs(1).TabStops(1) _  
        .Alignment = wdAlignTabCenter  
End Sub
```



AllCaps Property

-

True if the font is formatted as all capital letters. Returns **True**, **False**, or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle** (reverses the current setting). Read/write **Long**.

Remarks

Setting **AllCaps** to **True** sets **SmallCaps** to **False**, and vice versa.

Example

This example checks the third paragraph in the active document for text formatted as all capital letters.

```
If ActiveDocument.Paragraphs(3).Range.Font.AllCaps = True Then
    MsgBox "Text is all caps."
Else
    MsgBox "Text is not all caps."
End if
```

This example formats the selected text as all capital letters.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.AllCaps = True
Else
    MsgBox "You need to select some text."
End If
```



AllowAccentedUppercase Property

-
True if accents are retained when a French language character is changed to uppercase. Read/write **Boolean**.

Remarks

This property affects only text that's been marked as standard French. For all other languages, accents are always retained even if the **AllowAccentedUppercase** property is set to **False**.

If you change a character back to lowercase after an accent mark has been stripped from it, the accent won't reappear.

Example

This example sets Word to remove accent marks when characters in French text are changed to uppercase.

```
Options.AllowAccentedUppercase = False
```

This example returns the status of the **Allow accented uppercase in French** option on the **Edit** tab in the **Options** dialog box.

```
Dim blnUppercaseAccents as Boolean
```

```
blnUppercaseAccents = Options.AllowAccentedUppercase
```



AllowAutoFit Property

-
Allows Microsoft Word to automatically resize cells in a table to fit their contents. Read/write **Boolean**.

expression.**AllowAutoFit**

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

Example

This example sets the first table in the active document to automatically resize based on its contents.

```
Sub AllowFit()  
    ActiveDocument.Tables(1).AllowAutoFit = True  
End Sub
```



AllowBreakAcrossPage Property

-

Sets or returns a **Long** indicating whether lines in the rows of tables formatted with a specified style break across pages. **True** to break the lines in table rows across page breaks. **False** to keep the lines in a row of a table all on the same page. The default setting is **True**. Read/write.

expression.**AllowBreakAcrossPage**

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

Example

This example formats rows in tables formatted with the "Table Grid" style to not break at page breaks.

```
Sub DontSplitRows()  
    ActiveDocument.Styles("Table Grid") _  
        .Table.AllowBreakAcrossPage = False  
End Sub
```



AllowBreakAcrossPages Property

-
True if the text in a table row or rows are allowed to split across a page break. Can be **True**, **False** or **wdUndefined** (only some of the specified text is allowed to split). Read/write **Long**.

expression.**AllowBreakAcrossPages**

expression Required. An expression that returns a [TableStyle](#) object.

Example

This example creates a new document with a 5x5 table and prevents the third row of the table from being split during pagination.

```
Dim docNew As Document
Dim tableNew As Table

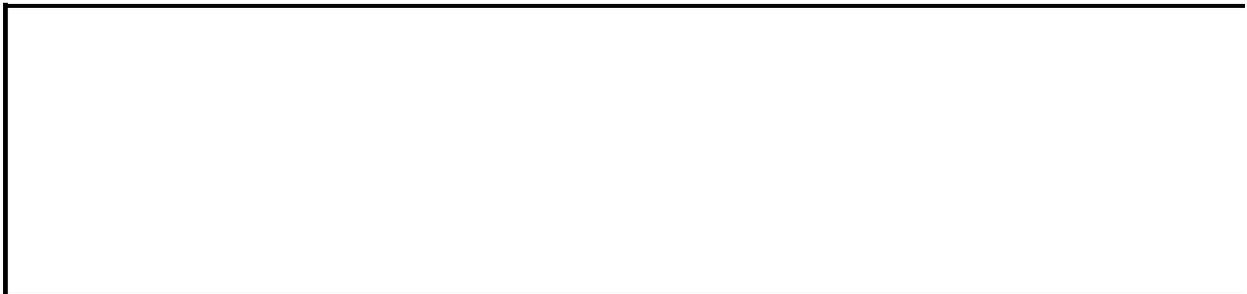
Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Range:=Selection.Range, _
    NumRows:=5, NumColumns:=5)

tableNew.Rows(3).AllowBreakAcrossPages = False
```

This example determines whether the rows in the current table can be split across pages. If the insertion point isn't in a table, a message box is displayed.

```
Dim lngAllowBreak as Long

Selection.Collapse Direction:=wdCollapseStart
If Selection.Tables.Count = 0 Then
    MsgBox "The insertion point is not in a table."
Else
    lngAllowBreak = Selection.Rows.AllowBreakAcrossPages
End If
```



AllowClickAndTypeMouse Property

True if Click and Type functionality is enabled. Read/write **Boolean**.

expression.AllowClickAndTypeMouse

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on Click and Type, see [About Click and Type](#).

Example

This example checks to determine whether Click and Type functionality is enabled. If it isn't enabled, the example sets this functionality based on the user's choice.

```
If Options.AllowClickAndTypeMouse = False Then
    x = MsgBox("Do you want to use Click and Type?", _
        vbYesNo)
    If x = vbYes Then
        Options.AllowClickAndTypeMouse = True
        MsgBox "Click and Type enabled!"
    End If
End If
```



AllowCombinedAuxiliaryForms Property

True if Microsoft Word ignores auxiliary verb forms when checking spelling in a Korean language document. Read/write **Boolean**.

expression.**AllowCombinedAuxiliaryForms**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example asks the user whether Microsoft Word should ignore auxiliary verb forms when checking spelling in a Korean language document.

```
If Options.AllowCombinedAuxiliaryForms = False Then
    x = MsgBox("Do you want to ignore auxiliary " _
        & "verb forms when checking spelling?", _
        vbYesNo)
    If x = vbYes Then
        Options.AllowCombinedAuxiliaryForms = True
        MsgBox "Auxiliary verb forms will be ignored!"
    End If
End If
```



AllowCompoundNounProcessing Property

True if Microsoft Word ignores compound nouns when checking spelling in a Korean language document. Read/write **Boolean**.

expression.**AllowCompoundNounProcessing**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example asks the user whether Microsoft Word should ignore compound nouns when checking spelling in a Korean language document.

```
If Options.AllowCompoundNounProcessing = False Then
    x = MsgBox("Do you want to ignore compound " _
        & "nouns when checking spelling?", _
        vbYesNo)
    If x = vbYes Then
        Options.AllowCompoundNounProcessing = True
        MsgBox "Compound nouns will be ignored!"
    End If
End If
```



AllowDragAndDrop Property

-
True if dragging and dropping can be used to move or copy a selection.
Read/write **Boolean**.

Example

This example turns on the drag-and-drop editing feature.

```
Options.AllowDragAndDrop = True
```

This example returns the status the **Drag-and-Drop text editing** option on the **Edit** tab in the **Options** dialog box.

```
Dim blnDragAndDrop as Boolean
```

```
blnDragAndDrop = Options.AllowDragAndDrop
```



AllowFastSave Property

-

True if Word saves only changes to a document. When reopening the document, Word uses the saved changes to reconstruct the document. Read/write **Boolean**.

Remarks

The **AllowFastSave** and **CreateBackup** properties cannot be set to **True** concurrently.

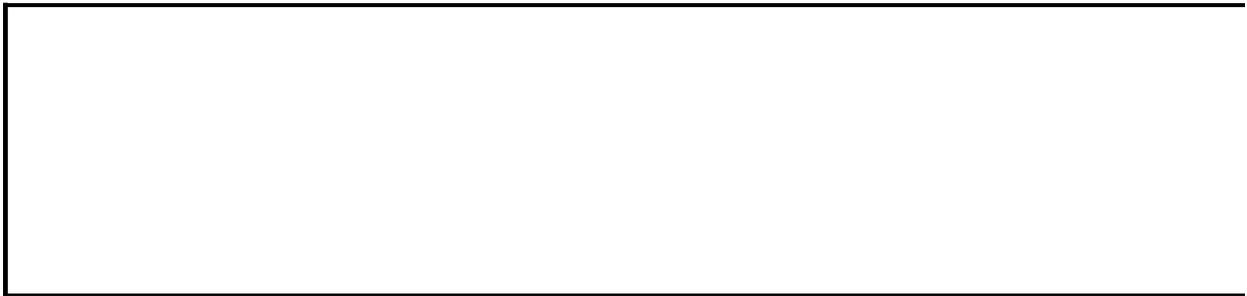
Example

This example sets Word to save the complete document, and then it saves the active document.

```
Options.AllowFastSave = False  
ActiveDocument.Save
```

This example returns the current status of the **Allow fast saves** option on the **Save** tab in the **Options** dialog box.

```
Dim blnFastSave as Boolean  
  
blnFastSave = Options.AllowFastSave
```



AllowOverlap Property

-

Rows object: Returns or sets a value that specifies whether the specified rows can overlap other rows. Returns **wdUndefined** if the specified rows include both overlapping rows and nonoverlapping rows. Can be set to either **True** or **False**. Read/write **Long**. Setting **AllowOverlap** to **True** also sets [WrapAroundText](#) to **True**, and setting **WrapAroundText** to **False** also sets **AllowOverlap** to **False**.

WrapFormat object: Returns or sets a value that specifies whether a given shape can overlap other shapes. Can be set to either **True** or **False**. Read/write **Long**.

Remarks

Because HTML doesn't support overlapping tables or shapes, **AllowOverlap** is ignored in Web layout view.

Example

This example specifies that text wraps around the selected table and that the table doesn't overlap any other wrapped tables.

```
Selection.Rows.WrapAroundText = True  
Selection.Rows.AllowOverlap = False
```

This example specifies that the first shape in the active document can overlap other shapes.

```
ActiveDocument.Shapes(1).WrapFormat.AllowOverlap = True
```



AllowPageBreaks Property

-
Allows Microsoft Word to break the specified table across pages. Read/write **Boolean**.

expression.**AllowPageBreaks**

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

Example

This example sets the second table in the active document to break across pages.

```
Sub BreakRow()  
    ActiveDocument.Tables(2).AllowPageBreaks = True  
End Sub
```



AllowPixelUnits Property

-
True if Microsoft Word uses pixels as the default unit of measurement for HTML features that support measurements. Read/write **Boolean**.

Example

This example sets Word to allow pixels as the default unit of measurement for HTML features.

```
Options.AllowPixelUnits = True
```



AllowPNG Property

-

True if PNG (Portable Network Graphics) is allowed as an image format when you save a document as a Web page. **False** if PNG is not allowed as an output format. The default value is **False**. Read/write **Boolean**.

Remarks

If you save images in the PNG format and if the Web browsers you are targeting support the PNG format, you might improve the image quality or reduce the size of those image files, and therefore decrease the download time.

Example

This example enables PNG as an output format for the active document.

```
ActiveDocument.WebOptions.AllowPNG = True
```

Alternatively, PNG can be enabled as the global default for the application for newly created documents.

```
Application.DefaultWebOptions.AllowPNG = True
```



↳ [Show All](#)

AlternativeText Property

-
Returns or sets the [alternative text](#) associated with a shape in a Web page.
Read/write **String**.

Example

The following example sets the alternative text for the selected shape in the active window. The selected shape is a picture of a mallard duck.

```
ActiveWindow.Selection.ShapeRange _  
    .AlternativeText = "This is a mallard duck."
```



AlwaysInFront Property

-
True if page borders are displayed in front of the document text. Read/write **Boolean**.

Example

This example adds a graphical page border in front of text in the first section in the active document.

```
Dim borderLoop as Border

With ActiveDocument.Sections(1)
    .Borders.AlwaysInFront = True
    For Each borderLoop In .Borders
        With borderLoop
            .ArtStyle = wdArtPeople
            .ArtWidth = 15
        End With
    Next borderLoop
End With
```



AlwaysSaveInDefaultEncoding Property

-

True if the default encoding is used when you save a Web page or plain text document, independent of the file's original encoding when opened. **False** if the original encoding of the file is used. The default value is **False**. Read/write **Boolean**.

Remarks

The [Encoding](#) property can be used to set the default encoding.

Example

This example sets the encoding to the default encoding. The encoding is used when you save the document as a Web page.

```
Application.DefaultWebOptions _  
    .AlwaysSaveInDefaultEncoding = True
```



Anchor Property

-

Returns a **Range** object that represents the anchoring range for the specified shape or shape range. Read-only.

Remarks

All **Shape** objects are anchored to a range of text but can be positioned anywhere on the page that contains the anchor. If you specify the anchoring range when you create a shape, the anchor is positioned at the beginning of the first paragraph that contains the anchoring range. If you don't specify the anchoring range, the anchoring range is selected automatically and the shape is positioned relative to the top and left edges of the page.

The shape will always remain on the same page as its anchor. If the **LockAnchor** property for the shape is set to **True**, you cannot drag the anchor from its position on the page.

If you use this property on a **ShapeRange** object that contains more than one shape, an error occurs.

Example

This example selects the paragraph that the first shape in the active document is anchored to.

```
ActiveDocument.Shapes(1).Anchor.Paragraphs(1).Range.Select
```



Angle Property

Returns or sets 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](#).

MsoCalloutAngleType can be one of these MsoCalloutAngleType constants.

msoCalloutAngle45

msoCalloutAngle90

msoCalloutAngleMixed

msoCalloutAngle30

msoCalloutAngle60

msoCalloutAngleAutomatic

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.

Example

This example sets the callout angle to 90 degrees for a callout named "co1" on the active document.

```
ActiveDocument.Shapes("co1").Callout.Angle = msoCalloutAngle90
```



AnimateScreenMovements Property

-

True if Word animates mouse movements, uses animated cursors, and animates actions such as background saving and find and replace operations. Read/write **Boolean**.

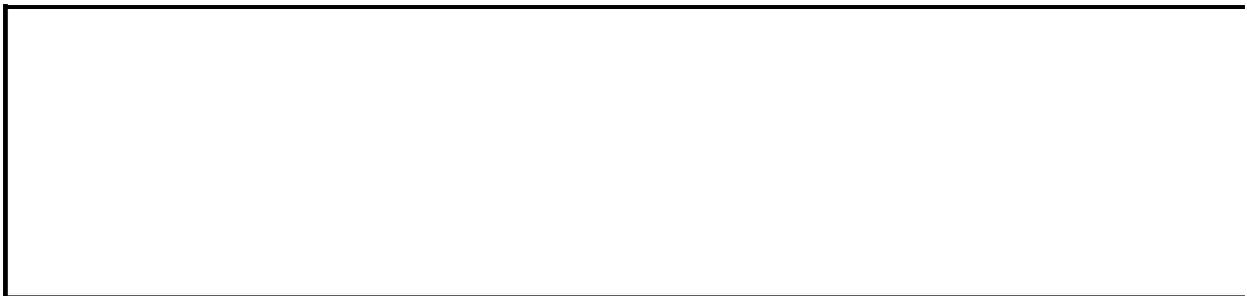
Example

This example sets Word to animate movements on the screen.

```
Options.AnimateScreenMovements = True
```

This example returns the current status of the **Provide feedback with animation** option on the **General** tab in the **Options** dialog box (**Tools** menu).

```
Dim blnAnimation as Boolean blnAnimation = Options.AnimateScreenMove
```



Animation Property

Returns or sets the type of animation applied to the font. Read/write [WdAnimation](#).

WdAnimation can be one of these WdAnimation constants.

wdAnimationBlinkingBackground

wdAnimationLasVegasLights

wdAnimationMarchingRedAnts

wdAnimationShimmer

wdAnimationMarchingBlackAnts

wdAnimationNone

wdAnimationSparkleText

expression.**Animation**

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

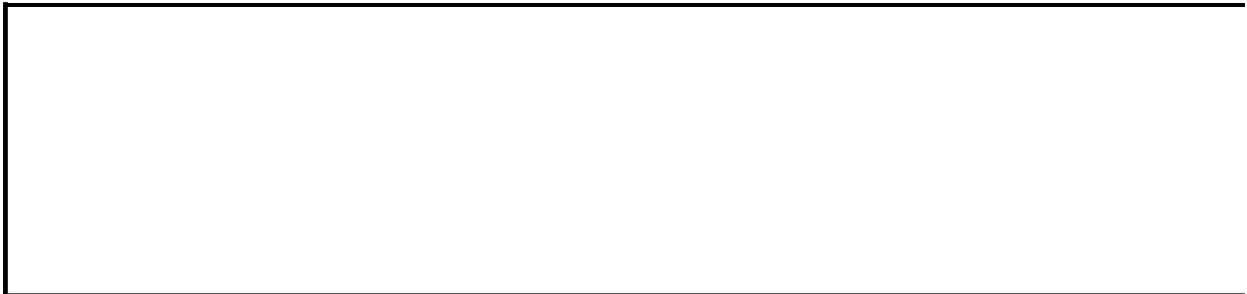
Example

This example animates the text in a new document.

```
Dim docNew As Document
Set docNew = Documents.Add
With docNew.Content
    .InsertAfter "This is a test of animation."
    .Font.Animation = wdAnimationLasVegasLights
End With
```

This example animates the selected text.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Animation = wdAnimationShimmer
Else
    MsgBox "You need to select some text."
End If
```



AnswerWizard Property

Returns an [AnswerWizard](#) object that contains the files used by the online Help search engine.

expression.**AnswerWizard**

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

Example

This example resets the Answer Wizard file list.

```
Sub AnswerWizardReset()  
    Application.AnswerWizard.ResetFileList  
End Sub
```



AntonymList Property

Returns a list of antonyms for the word or phrase. The list is returned as an array of strings. Read-only **Variant**.

expression.**AntonymList**

expression Required. An expression that returns a [SynonymInfo](#) object.

Remarks

The **AntonymList** property is a property of the **SynonymInfo** object, which can be returned from either a range or the application. If this object is returned from the application, you specify the word to look up and the language to use. When the object is returned from a range, the range is looked up using the language of the range.

Example

This example returns a list of antonyms for the word "big" in U.S. English.

```
Dim arrayAntonyms As Variant
Dim intLoop As Integer

arrayAntonyms = SynonymInfo(Word:="big", _
    LanguageID:=wdEnglishUS).AntonymList
For intLoop = 1 To UBound(arrayAntonyms)
    MsgBox arrayAntonyms(intLoop)
Next intLoop
```

This example returns a list of antonyms for the word or phrase in the selection and displays them in the **Immediate** window in the Visual Basic Editor.

```
Dim arrayAntonyms As Variant
Dim intLoop As Integer

arrayAntonyms = Selection.Range.SynonymInfo.AntonymList
If UBound(arrayAntonyms) <> 0 Then
    For intLoop = 1 To UBound(arrayAntonyms)
        Debug.Print arrayAntonyms(intLoop) & Str(intLoop)
    Next intLoop
Else
    MsgBox "No antonyms were found."
End If
```

This example returns a list of antonyms, if there are any, for the third word in the active document.

```
Dim rngTemp As Range
Dim arrayAntonyms As Variant
Dim intLoop As Integer

Set rngTemp = ActiveDocument.Words(3)

arrayAntonyms = rngTemp.SynonymInfo.AntonymList
If UBound(arrayAntonyms) = 0 Then
    MsgBox "There are no antonyms for the third word."
Else
    For intLoop = 1 To UBound(arrayAntonyms)
        MsgBox arrayAntonyms(intLoop)
    Next intLoop
End If
```

```
    Next intLoop  
End If
```



Application Property

-

Used without an object qualifier, this property returns an [Application](#) object that represents the Microsoft Word application. 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.

expression.**Application**

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

Remarks

Visual Basic's **CreateObject** and **GetObject** functions give you access to an OLE Automation object from a Visual Basic for Applications project.

Example

This example displays scroll bars, screen tips, and the status bar for Microsoft Word.

```
With Application  
    .DisplayScrollBars = True  
    .DisplayScreenTips = True  
    .DisplayStatusBar = True  
End With
```

This example displays the Microsoft Excel startup path if Excel is running.

```
If Tasks.Exists(Name:="Microsoft Excel") = True Then  
    Set myobject = GetObject("", "Excel.Application")  
    MsgBox myobject.Application.StartupPath  
    Set myobject = Nothing  
End If
```



ApplyFarEastFontsToAscii Property

True if Microsoft Word applies East Asian fonts to Latin text. Read/write **Boolean**.

expression.**ApplyFarEastFontsToAscii**

expression Required. An expression that returns an [Options](#) object.

Remarks

This property applies only when you have selected an East Asian language for editing. If this property is **False** and you apply an East Asian font to a specified range, Word will not apply the font to any Latin text in the range.

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets Microsoft Word to apply East Asian fonts to Latin text.

```
Options.ApplyFarEastFontsToAscii = True
```



ApplyStyleFirstColumn Property

True for Microsoft Word to apply first-column formatting to the first column of the specified table. Read/write **Boolean**.

expression.**ApplyStyleFirstColumn**

expression Required. An expression that returns a [Table](#) object.

Remarks

The specified table style must contain first-column formatting in order to apply this formatting to a table.

Example

This example formats the second table in the active document with the table style "Table Style 1" and removes formatting for the first and last rows and the first and last columns. This example assumes that a table style named "Table Style 1" exists and that it contains first column formatting.

```
Sub TableStyles()  
  With ActiveDocument.Tables(2)  
    .Style = "Table Style 1"  
    .ApplyStyleFirstColumn = False  
    .ApplyStyleHeadingRows = False  
    .ApplyStyleLastColumn = False  
    .ApplyStyleLastRow = False  
  End With  
End Sub
```



ApplyStyleHeadingRows Property

True for Microsoft Word to apply heading-row formatting to the first row of the selected table. Read/write **Boolean**.

expression.**ApplyStyleHeadingRows**

expression Required. An expression that returns a [Table](#) object.

Remarks

The specified table style must contain heading-row formatting in order to apply this formatting to a table.

Example

This example formats the second table in the active document with the table style "Table Style 1" and removes formatting for the first and last rows and the first and last columns. This example assumes that a table style named "Table Style 1" exists and that it contains heading-row formatting.

```
Sub TableStyles()  
  With ActiveDocument.Tables(2)  
    .Style = "Table Style 1"  
    .ApplyStyleFirstColumn = False  
    .ApplyStyleHeadingRows = False  
    .ApplyStyleLastColumn = False  
    .ApplyStyleLastRow = False  
  End With  
End Sub
```



ApplyStyleLastColumn Property

True for Microsoft Word to apply last-column formatting to the last column of the specified table. Read/write **Boolean**.

expression.**ApplyStyleLastColumn**

expression Required. An expression that returns a [Table](#) object.

Remarks

The specified table style must contain last-column formatting in order to apply this formatting to a table.

Example

This example formats the second table in the active document with the table style "Table Style 1" and removes formatting for the first and last rows and the first and last columns. This example assumes that a table style named "Table Style 1" exists and that it contains last-column formatting.

```
Sub TableStyles()  
  With ActiveDocument.Tables(2)  
    .Style = "Table Style 1"  
    .ApplyStyleFirstColumn = False  
    .ApplyStyleHeadingRows = False  
    .ApplyStyleLastColumn = False  
    .ApplyStyleLastRow = False  
  End With  
End Sub
```



ApplyStyleLastRow Property

True for Microsoft Word to apply last-row formatting to the last row of the specified table. Read/write **Boolean**.

expression.**ApplyStyleLastRow**

expression Required. An expression that returns a [Table](#) object.

Remarks

The specified table style must contain last-row formatting in order to apply this formatting to a table.

Example

This example formats the second table in the active document with the table style "Table Style 1" and removes formatting for the first and last rows and the first and last columns. This example assumes that a table style named "Table Style 1" exists and that it contains last-row formatting.

```
Sub TableStyles()  
    With ActiveDocument.Tables(2)  
        .Style = "Table Style 1"  
        .ApplyStyleFirstColumn = False  
        .ApplyStyleHeadingRows = False  
        .ApplyStyleLastColumn = False  
        .ApplyStyleLastRow = False  
    End With  
End Sub
```



↳ [Show All](#)

ArabicMode Property

Returns or sets the mode for the Arabic spelling checker. Read/write [WdAraSpeller](#).

WdAraSpeller can be one of these WdAraSpeller constants.

wdBoth The spelling checker uses spelling rules regarding both Arabic words ending with the letter *yaa* and Arabic words beginning with an *alef hamza*.

wdInitialAlef The spelling checker uses spelling rules regarding Arabic words beginning with an *alef hamza*.

wdFinalYaa The spelling checker uses spelling rules regarding Arabic words ending with the letter *yaa*.

wdNone The spelling checker ignores spelling rules regarding either Arabic words ending with the letter *yaa* or Arabic words beginning with an *alef hamza*.

expression.**ArabicMode**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the spelling checker to ignore spelling rules regarding Arabic words beginning with an *alef hamza*.

```
Options.ArabicMode = wdInitialAlef
```



↳ [Show All](#)

ArabicNumeral Property

Returns or sets the numeral style for an Arabic language document. Read/write [WdArabicNumeral](#).

WdArabicNumeral can be one of these WdArabicNumeral constants.

wdNumeralArabic

wdNumeralHindi

wdNumeralContext

wdNumeralSystem

expression.**ArabicNumeral**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the numeral style to Hindi.

```
Options.ArabicNumeral = wdNumeralHindi
```



ArtStyle Property

Returns or sets the graphical page-border design for a document. Read/write [WdPageBorderArt](#).

WdPageBorderArt can be one of these WdPageBorderArt constants.

wdArtSeattle

wdArtSharksTeeth

wdArtSkyrocket

wdArtSnowflakes

wdArtSouthwest

wdArtStars3D

wdArtStarsShadowed

wdArtSun

wdArtTornPaper

wdArtTrees

wdArtTriangles

wdArtTribal2

wdArtTribal4

wdArtTribal6

wdArtTwistedLines2

wdArtWaveline

wdArtWeavingBraid

wdArtWeavingStrips

wdArtWoodwork

wdArtZanyTriangles

wdArtZigZagStitch

wdArtCirclesLines

wdArtClassicalWave

wdArtCompass

wdArtConfettiGrays
wdArtConfettiStreamers
wdArtCornerTriangles
wdArtCouponCutoutDots
wdArtCreaturesButterfly
wdArtCreaturesInsects
wdArtScaredCat
wdArtShadowedSquares
wdArtShorebirdTracks
wdArtSnowflakeFancy
wdArtSombrero
wdArtStars
wdArtStarsBlack
wdArtStarsTop
wdArtSwirligig
wdArtTornPaperBlack
wdArtTriangleParty
wdArtTribal1
wdArtTribal3
wdArtTribal5
wdArtTwistedLines1
wdArtVine
wdArtWeavingAngles
wdArtWeavingRibbon
wdArtWhiteFlowers
wdArtXIllusions
wdArtZigZag
wdArtChristmasTree
wdArtCirclesRectangles
wdArtClocks
wdArtConfetti
wdArtConfettiOutline
wdArtConfettiWhite

wdArtCouponCutoutDashes
wdArtCrazyMaze
wdArtCreaturesFish
wdArtCreaturesLadyBug
wdArtCrossStitch
wdArtCup
wdArtDecoArch
wdArtDecoArchColor
wdArtDecoBlocks
wdArtDiamondsGray
wdArtDoubleD
wdArtDoubleDiamonds
wdArtEarth1
wdArtEarth2
wdArtEclipsingSquares1
wdArtEclipsingSquares2
wdArtEggsBlack
wdArtFans
wdArtFilm
wdArtFirecrackers
wdArtFlowersBlockPrint
wdArtFlowersDaisies
wdArtFlowersModern1
wdArtFlowersModern2
wdArtFlowersPansy
wdArtFlowersRedRose
wdArtFlowersRoses
wdArtFlowersTeacup
wdArtFlowersTiny
wdArtGems
wdArtGingerbreadMan
wdArtGradient
wdArtHandmade1

wdArtHandmade2
wdArtHeartBalloon
wdArtHeartGray
wdArtHearts
wdArtHeebieJeebies
wdArtHolly
wdArtHouseFunky
wdArtHypnotic
wdArtIceCreamCones
wdArtLightBulb
wdArtLightning1
wdArtLightning2
wdArtMapleLeaf
wdArtMapleMuffins
wdArtMapPins
wdArtMarquee
wdArtMarqueeToothed
wdArtMoons
wdArtMosaic
wdArtMusicNotes
wdArtNorthwest
wdArtOvals
wdArtPackages
wdArtPalmsBlack
wdArtPalmsColor
wdArtPaperClips
wdArtPapyrus
wdArtPartyFavor
wdArtPartyGlass
wdArtPencils
wdArtPeople
wdArtPeopleHats
wdArtPeopleWaving

wdArtPoinsettias
wdArtPostageStamp
wdArtPumpkin1
wdArtPushPinNote1
wdArtPushPinNote2
wdArtPyramids
wdArtPyramidsAbove
wdArtQuadrants
wdArtRings
wdArtSafari
wdArtSawtooth
wdArtSawtoothGray
wdArtApples
wdArtArchedScallops
wdArtBabyPacifier
wdArtBabyRattle
wdArtBalloons3Colors
wdArtBalloonsHotAir
wdArtBasicBlackDashes
wdArtBasicBlackDots
wdArtBasicBlackSquares
wdArtBasicThinLines
wdArtBasicWhiteDashes
wdArtBasicWhiteDots
wdArtBasicWhiteSquares
wdArtBasicWideInline
wdArtBasicWideMidline
wdArtBasicWideOutline
wdArtBats
wdArtBirds
wdArtBirdsFlight
wdArtCabins
wdArtCakeSlice

wdArtCandyCorn
wdArtCelticKnotwork
wdArtCertificateBanner
wdArtChainLink
wdArtChampagneBottle
wdArtCheckedBarBlack
wdArtCheckedBarColor
wdArtCheckered

expression.**ArtStyle**

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

Example

This example adds a border of black dots around each page in first section in the selection.

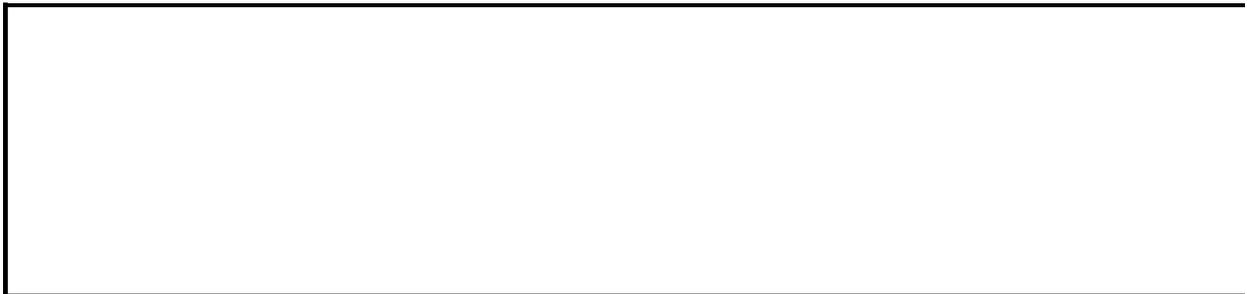
```
Dim borderLoop As Border

For Each borderLoop In Selection.Sections(1).Borders
    With borderLoop
        .ArtStyle = wdArtBasicBlackDots
        .ArtWidth = 6
    End With
Next borderLoop
```

This example adds a picture border around each page in section one in the active document.

```
Dim borderLoop As Border

With ActiveDocument.Sections(1)
    .Borders.AlwaysInFront = True
    For Each borderLoop In .Borders
        With borderLoop
            .ArtStyle = wdArtPeople
            .ArtWidth = 15
        End With
    Next borderLoop
End With
```



ArtWidth Property

-
Returns or sets the width (in points) of the specified graphical page border.
Read/write **Long**.

Example

This example adds a 6-point dotted border around each page in the first section in the selection.

```
Dim borderLoop As Border
```

```
For Each borderLoop In Selection.Sections(1).Borders
```

```
    With borderLoop
```

```
        .ArtStyle = wdArtBasicBlackDots
```

```
        .ArtWidth = 6
```

```
    End With
```

```
Next borderLoop
```



Assistant Property

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.

```
Assistant.Visible = True
```

This example displays the Office Assistant and moves it to the upper-left region of the screen.

```
With Assistant  
    .Visible = True  
    .Move xLeft:=100, yTop:=100  
End With
```

This example displays the Office Assistant with a custom message in a balloon.

```
With Assistant  
    .Visible = True  
    Set bln = .NewBalloon  
    With bln  
        .Mode = msoModeAutoDown  
        .Text = "Hello"  
        .Button = msoButtonSetNone  
        .Show  
    End With  
End With
```



AttachedTemplate Property

-

Returns a [Template](#) object that represents the template attached to the specified document. To set this property, specify either the name of the template or an expression that returns a **Template** object. Read/write **Variant**.

Example

This example displays the name and path of the template attached to the active document.

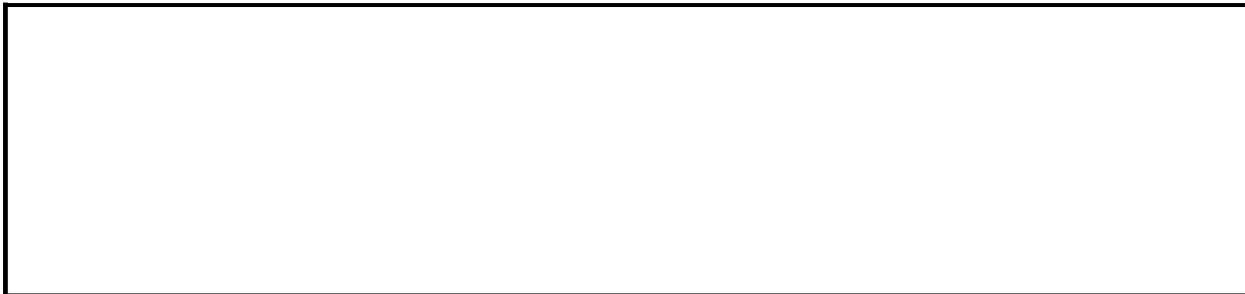
```
Set myTemplate = ActiveDocument.AttachedTemplate  
MsgBox myTemplate.Path & Application.PathSeparator _  
    & myTemplate.Name
```

This example inserts the contents of the Spike (a built-in AutoText entry) at the beginning of document one.

```
Set myRange = Documents(1).Range(0, 0)  
Documents(1).AttachedTemplate.AutoTextEntries("Spike") _  
    .Insert myRange
```

This example attaches the template "Letter.dot" to the active document.

```
ActiveDocument.AttachedTemplate = "C:\Templates\Letter.dot"
```



AttentionLine Property

-
Returns or sets the attention line text for a letter created by the Letter Wizard.
Read/write **String**.

Example

This example retrieves the Letter Wizard elements from the active document. If the attention line isn't blank, the example displays the text in a message box.

```
If ActiveDocument.GetLetterContent.AttentionLine <> "" Then  
    MsgBox ActiveDocument.GetLetterContent.AttentionLine  
End If
```

This example retrieves the Letter Wizard elements from the active document, changes the attention line text, and then uses the **SetLetterContent** method to update the document to reflect the changes.

```
Dim lcTemp As LetterContent  
  
Set lcTemp = ActiveDocument.GetLetterContent  
  
lcTemp.AttentionLine = "Greetings"  
ActiveDocument.SetLetterContent LetterContent:=lcTemp
```



↳ [Show All](#)

Author Property

-
▶ [Author property as it applies to the **Comment** object.](#)

Returns or sets the author name for a comment. Read/write **String**.

expression.**Author**

expression Required. An expression that returns one of the above objects.

Remarks

Changing the author for one comment will change the author for all comments in a document.

▶ [Author property as it applies to the **Revision** object.](#)

Returns the name of the user who made the specified tracked change. Read-only **String**.

expression.**Author**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Comment** object.](#)

This example sets the author name and initials for the first comment in the active document.

```
If ActiveDocument.Comments.Count >= 1 Then
    With ActiveDocument.Comments(1)
        .Author = "Joe Smith"
        .Initial = "JAS"
    End With
End If
```

This example returns the author name for the first comment in the selection.

```
Dim strAuthor as String

If Selection.Comments.Count >= 1 Then _
    strAuthor = Selection.Comments(1).Author
```

▶ [As it applies to the **Revision** object.](#)

This example displays the author name for the first tracked change in the first selected section.

```
Dim rngSection as Range

Set rngSection = Selection.Sections(1).Range
MsgBox "Revisions made by " & rngSection.Revisions(1).Author
```



AutoAdjustRightIndent Property

-

True if Microsoft Word is set to automatically adjust the right indent for the specified paragraphs if you've specified a set number of characters per line. Returns **wdUndefined** if the **AutoAdjustRightIndent** property is set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example sets Microsoft Word to automatically adjust the right indent for the selected paragraphs if you've specified a set number of characters per line.

```
With Selection.ParagraphFormat  
    .AutoAdjustRightIndent = True  
End With
```



AutoAttach Property

-
True if 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 to) is to the left or right of the callout text box. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

Remarks

When the value of this property is **True**, 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 **False**, 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 active document. If you drag the text box for each of these callouts to the left of the callout line origin, the place on the text box where the callout line attaches will change for the automatically attached callout.

```
Dim docActive as Document

Set docActive = ActiveDocument

With docActive.Shapes
    With .AddCallout(msoCalloutTwo, 100, 170, 200, 50)
        .TextFrame.TextRange.Text = "auto-attached"
        .Callout.AutoAttach = msoTrue
    End With
    With .AddCallout(msoCalloutTwo, 100, 350, 200, 50)
        .TextFrame.TextRange.Text = "not auto-attached"
        .Callout.AutoAttach = msoFalse
    End With
End With
```



AutoCaptions Property

Returns an [AutoCaptions](#) collection that represents the captions that are automatically added when items such as tables and pictures are inserted into a document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the name of each item that automatically gets a caption when inserted into the document.

```
Dim captionLoop as AutoCaption
```

```
For Each captionLoop In AutoCaptions
```

```
    If captionLoop.AutoInsert Then MsgBox captionLoop.Name
```

```
Next captionLoop
```



AutoCorrect Property

-
Returns an [AutoCorrect](#) object that contains the current AutoCorrect options, entries, and exceptions. Read-only.

Example

This example adds an AutoCorrect replacement entry. After this code runs, every instance of "sr" that's typed in a document will automatically be replaced with "Stella Richards."

```
AutoCorrect.Entries.Add Name:= "sr", Value:= "Stella Richards"
```

This example deletes the specified AutoCorrect entry if it exists.

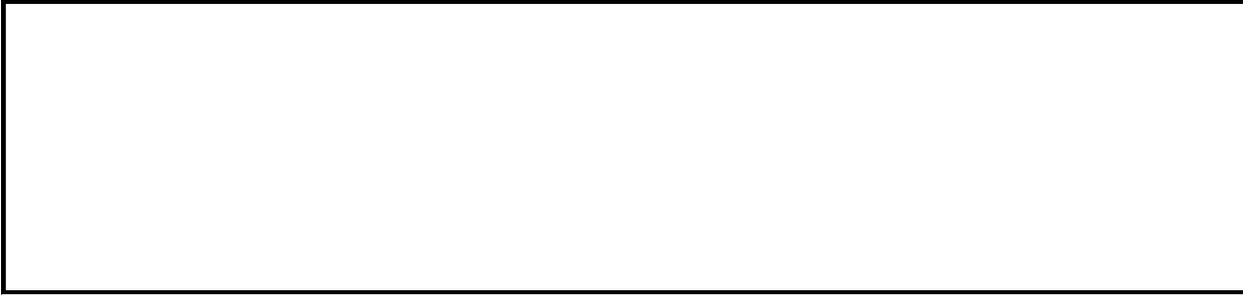
```
Dim strInput as String
Dim aceLoop as AutoCorrectEntry
Dim blnMatch as Boolean
Dim intConfirm as Integer

blnMatch = False

strInput = InputBox("Enter the AutoCorrect entry to delete.")

For Each aceLoop in AutoCorrect.Entries
    With aceLoop
        If .Name = strInput Then
            blnMatch = True
            intConfirm = _
                MsgBox("Are you sure you want to delete " & _
                    .Name, 4)
            If intConfirm = vbYes Then
                .Delete
            End If
        End If
    End With
Next aceLoop

If blnMatch <> True Then
    MsgBox "There was no AutoCorrect entry: " & strInput
End If
```



AutoCorrectEmail Property

Returns an **AutoCorrect** object that represents automatic corrections made to e-mail messages.

expression.**AutoCorrectEmail**

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

Example

This example adds AutoCorrect entries for e-mail messages. After this code runs, every instance of "allways," "hte," and "hwen" that's typed in an e-mail message will be replaced with "always," "the," and "when," respectively.

```
Sub AutoCorrectEmailAddress()  
  With Application.AutoCorrectEmail  
    .Entries.Add Name:="allways", Value:="always"  
    .Entries.Add Name:="hte", Value:="the"  
    .Entries.Add Name:="hwen", Value:="when"  
  End With  
End Sub
```



↳ [Show All](#)

AutoCreateNewDrawings Property

True for Microsoft Word to draw newly created shapes in a [drawing canvas](#).
Read/write **Boolean**.

expression.**AutoCreateNewDrawings**

expression Required. An expression that returns an [Options](#) object.

Remarks

The **AutoCreateNewDrawings** property only affects shapes as they are added from within Word. If shapes are added through Visual Basic for Applications code, they are added as specified in the code regardless of whether this option is set to **True** or **False**.

Example

This example sets Word to add newly created shapes directly to the document and not within a drawing canvas.

```
Sub NewDrawings()  
    Application.Options.AutoCreateNewDrawings = False  
End Sub
```



↳ [Show All](#)

AutoFormat Property

-
Sets or returns an [MsoTriState](#) constant specifying the automatic formatting state for a diagram. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse Disables automatic formatting.

msoTriStateMixed Not used for this property.

msoTriStateToggle Not used for this property.

msoTrue Formats a diagram to format automatically.

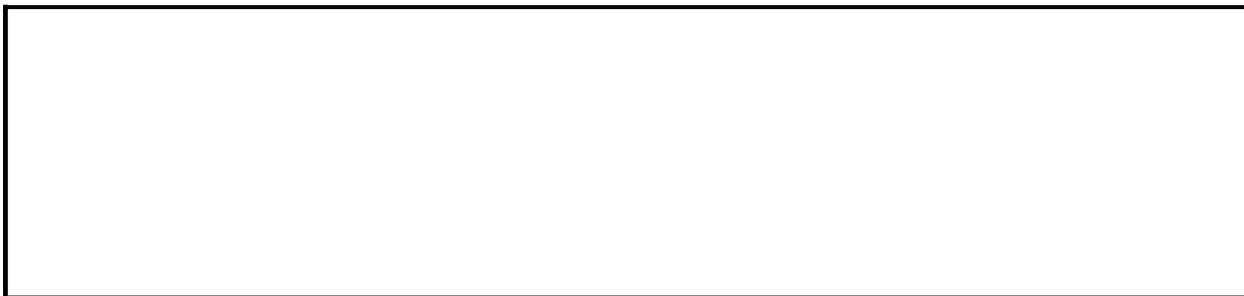
expression.**AutoFormat**

expression Required. An expression that returns a [Diagram](#) object.

Example

This example creates a diagram in the current document and turns on automatic formatting for the diagram.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add a pyramid diagram to current document and first child node  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child node  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    'Enable automatic formatting for the diagram and convert  
    'it to a radial diagram  
    With dgnNode.Diagram  
        .AutoFormat = msoTrue  
        .Convert Type:=msoDiagramRadial  
    End With  
  
End Sub
```



AutoFormatApplyBulletedLists Property

True if characters (such as asterisks, hyphens, and greater-than signs) at the beginning of list paragraphs are replaced with bullets from the **Bullets and Numbering** dialog box (**Format** menu) when Word formats a document or range automatically. Read/write **Boolean**.

Example

This example replaces any characters used at the beginning of list paragraphs in the current selection with bullets.

```
Options.AutoFormatApplyBulletedLists = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Automatic bulleted lists** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (Tools menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatApplyBulletedLists
```



AutoFormatApplyFirstIndents Property

True if Microsoft Word replaces a space entered at the beginning of a paragraph with a first-line indent when Word formats a document or range automatically.
Read/write **Boolean**.

expression.**AutoFormatApplyFirstIndents**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to replace a space entered at the beginning of a paragraph with a first-line indent and automatically formats the selected range.

```
Options.AutoFormatApplyFirstIndents = True  
Selection.Range.AutoFormat
```



AutoFormatApplyHeadings Property

-

True if styles are automatically applied to headings when Word formats a document or range automatically. Read/write **Boolean**.

Example

This example applies the Heading 1 through Heading 9 styles to headings in the current selection.

```
Options.AutoFormatApplyHeadings = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Headings** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean  
  
blnAutoFormat = Options.AutoFormatApplyHeadings
```



AutoFormatApplyLists Property

-
True if styles are automatically applied to lists when Word formats a document or range automatically. Read/write **Boolean**.

Example

This example applies styles to any lists in the current selection.

```
Options.AutoFormatApplyLists = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Lists** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatApplyLists
```



AutoFormatApplyOtherParas Property

-

True if styles are automatically applied to paragraphs that aren't headings or list items when Word formats a document or range automatically. Read/write **Boolean**.

Example

This example automatically applies styles to paragraphs in the current selection.

```
Options.AutoFormatApplyOtherParas = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Other paragraphs** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean  
  
blnAutoFormat = Options.AutoFormatApplyOtherParas
```



AutoFormatAsYouTypeApplyBorders Property

True if a series of three or more hyphens (-), equal signs (=), or underscore characters (_) are automatically replaced by a specific border line when the ENTER key is pressed. Read/write **Boolean**.

Remarks

Hyphens (-) are replaced by a 0.75-point line, equal signs (=) are replaced by a 0.75-point double line, and underscore characters (_) are replaced by a 1.5-point line.

Example

This example causes sequences of three or more hyphens (-), equal signs (=), or underscore characters (_) to be transformed into borders.

```
Options.AutoFormatAsYouTypeApplyBorders = True
```

This example returns the current setting for the **Borders** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
MsgBox Options.AutoFormatAsYouTypeApplyBorders
```



AutoFormatAsYouTypeApplyBulletedProperty

True if bullet characters (such as asterisks, hyphens, and greater-than signs) are replaced with bullets from the **Bullets And Numbering** dialog box (**Format** menu) as you type. Read/write **Boolean**.

Example

This example causes characters to be replaced with bullets when typed in a list.

```
Options.AutoFormatAsYouTypeApplyBulletedLists = True
```

This example returns the status of the **Automatic bulleted lists** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeApplyBulletedLists
```



AutoFormatAsYouTypeApplyClosings Property

True for Microsoft Word to automatically apply the Closing style to letter closings as you type. Read/write **Boolean**.

expression.**AutoFormatAsYouTypeApplyClosings**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically apply the Closing style to letter closings as you type.

```
Sub AutoClosings()  
    Options.AutoFormatAsYouTypeApplyClosings = True  
End Sub
```



AutoFormatAsYouTypeApplyDates Property

True for Microsoft Word to automatically apply the Date style to dates as you type. Read/write.

expression.**AutoFormatAsYouTypeApplyDates**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically apply the Date style to dates as you type.

```
Sub AutoApplyDates()  
    Options.AutoFormatAsYouTypeApplyDates = True  
End Sub
```



AutoFormatAsYouTypeApplyFirstInd Property

True for Microsoft Word to automatically replace a space entered at the beginning of a paragraph with a first-line indent. Read/write.

expression.**AutoFormatAsYouTypeApplyFirstIndents**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically replace a space entered at the beginning of a paragraph with a first-line indent as you type.

```
Sub ApplyFirstIndents()  
    Options.AutoFormatAsYouTypeApplyFirstIndents = True  
End Sub
```



AutoFormatAsYouTypeApplyHeadingProperty

True if styles are automatically applied to headings as you type. Read/write **Boolean**.

Example

This example sets Word to automatically apply the Heading1 through Heading 9 styles to headings as you type.

```
Options.AutoFormatAsYouTypeApplyHeadings = True
```

This example returns the status of the **Headings** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeApplyHeadings
```



AutoFormatAsYouTypeApplyNumberProperty

True if paragraphs are automatically formatted as numbered lists with a numbering scheme from the **Bullets and Numbering** dialog box (**Format** menu), according to what's typed. For example, if a paragraph starts with "1.1" and a tab character, Word automatically inserts "1.2" and a tab character after the ENTER key is pressed. Read/write **Boolean**.

Example

This example causes lists to be automatically numbered as you type.

```
Options.AutoFormatAsYouTypeApplyNumberedLists = True
```

This example returns the status of the **Automatic numbered lists** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeApplyNumberedLists
```



AutoFormatAsYouTypeApplyTables Property

True if Word automatically creates a table when you type a plus sign, a series of hyphens, another plus sign, and so on, and then press ENTER. The plus signs become the column borders, and the hyphens become the column widths.
Read/write **Boolean**.

Example

This example sets Word to automatically create tables as you type.

```
Options.AutoFormatAsYouTypeApplyTables = True
```

This example returns the status of the **Tables** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeApplyTables
```



AutoFormatAsYouTypeAutoLetterWizard Property

True for Microsoft Word to automatically start the Letter Wizard when the user enters a letter salutation or closing. Read/write.

expression.**AutoFormatAsYouTypeAutoLetterWizard**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically start the Letter Wizard when the user enters a letter salutation or closing.

```
Sub AutoLetterWizard()  
    Options.AutoFormatAsYouTypeAutoLetterWizard = True  
End Sub
```



AutoFormatAsYouTypeDefineStyles Property

True if Word automatically creates new styles based on manual formatting.
Read/write **Boolean**.

Example

This example sets Word to automatically create styles as you type.

```
Options.AutoFormatAsYouTypeDefineStyles = True
```

This example returns the status of the **Define styles based on your formatting** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeDefineStyles
```



AutoFormatAsYouTypeDeleteAutoSpaces Property

True for Microsoft Word to automatically delete spaces inserted between Japanese and Latin text as you type. Read/write.

expression.**AutoFormatAsYouTypeDeleteAutoSpaces**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically delete spaces inserted between Japanese and Latin text as you type.

```
Sub AutoDeleteSpaces()  
    Options.AutoFormatAsYouTypeDeleteAutoSpaces = True  
End Sub
```



AutoFormatAsYouTypeFormatListItemProperty

True if Word repeats character formatting applied to the beginning of a list item to the next list item. Read/write **Boolean**.

Example

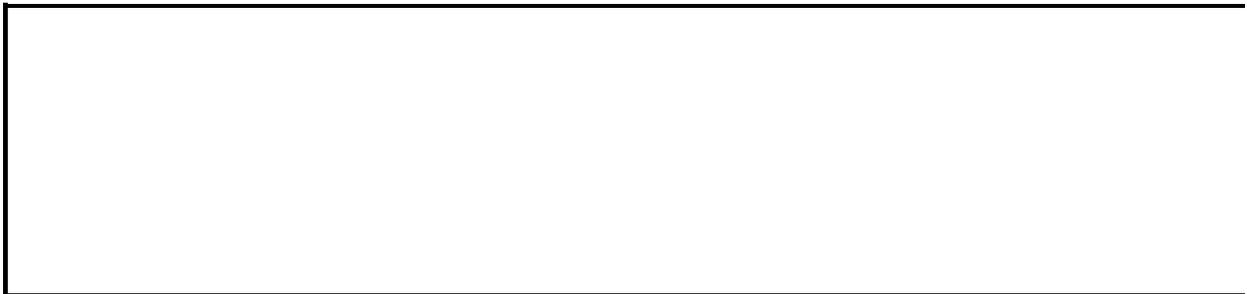
This example sets Word to automatically repeat character formatting at the beginning of list items.

```
Options.AutoFormatAsYouTypeFormatListItemBeginning = True
```

This example returns the status of the **Format beginning of list item like the one before it** option in the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Options** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = _  
    Options.AutoFormatAsYouTypeFormatListItemBeginning
```



AutoFormatAsYouTypeInsertClosings Property

True for Microsoft Word to automatically insert the corresponding memo closing when the user enters a memo heading. Read/write.

expression.**AutoFormatAsYouTypeInsertClosings**

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

Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically insert the corresponding memo closing when the user enters a memo heading.

```
Sub AutoInsertClosings()  
    Options.AutoFormatAsYouTypeInsertClosings = True  
End Sub
```



AutoFormatAsYouTypeInsertOvers Property

True for Microsoft Word to automatically insert " 以上 " when the user enters "記" or "案". Read/write **Boolean**.

expression.**AutoFormatAsYouTypeInsertOvers**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to automatically insert "以上" when the user enters "記" or "案".

```
Options.AutoFormatAsYouTypeInsertOvers = True
```



AutoFormatAsYouTypeMatchParenth Property

True for Microsoft Word to automatically correct improperly paired parentheses.
Read/write.

expression.**AutoFormatAsYouTypeMatchParentheses**

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

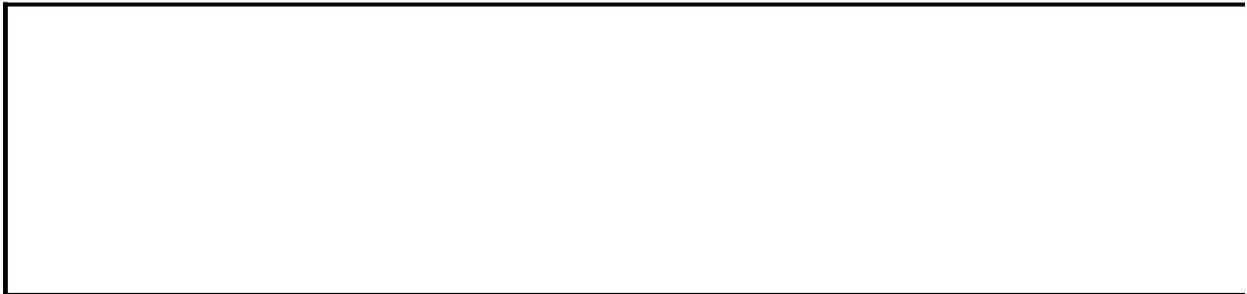
Remarks

For more information on Japanese AutoFormat options, see [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically correct improperly paired parentheses as you type.

```
Sub AutoMatchParentheses()  
    Options.AutoFormatAsYouTypeMatchParentheses = True  
End Sub
```



AutoFormatAsYouTypeReplaceFarEa Property

True for Microsoft Word to automatically correct long vowel sounds and dashes.
Read/write.

expression.**AutoFormatAsYouTypeReplaceFarEastDashes**

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

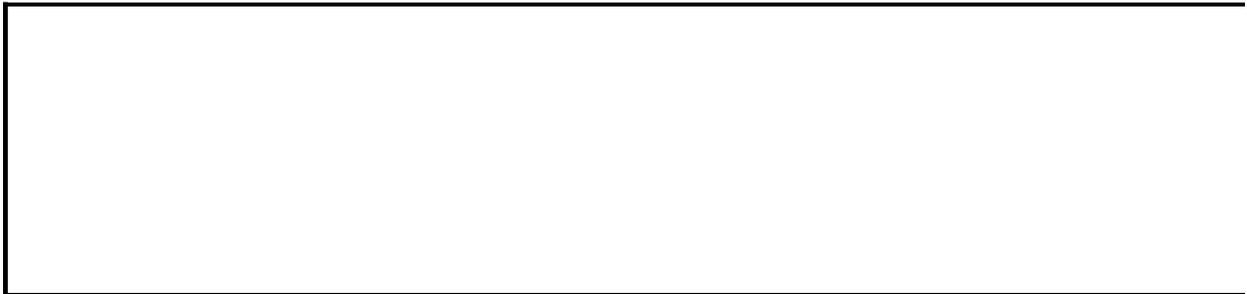
Remarks

For more information on Japanese AutoFormat options, [Automatically correct text as you type in another language](#).

Example

This example sets Microsoft Word to automatically correct long vowel sounds and dashes as you type.

```
Sub AutoFarEastDashes()  
    Options.AutoFormatAsYouTypeReplaceFarEastDashes = True  
End Sub
```



AutoFormatAsYouTypeReplaceFractionProperty

True if typed fractions are replaced with fractions from the current character set as you type. For example, "1/2" is replaced with "½." Read/write **Boolean**.

Example

This example turns off the automatic replacement of typed fractions.

```
Options.AutoFormatAsYouTypeReplaceFractions = False
```

This example returns the status of the **Fractions (1/2) with fraction character (½)** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeReplaceFractions
```



AutoFormatAsYouTypeReplaceHyperProperty

-
True if e-mail addresses, server and share names (also known as UNC paths), and Internet addresses (also known as URLs) are automatically changed to hyperlinks as you type. Read/write **Boolean**.

Remarks

Word changes any text that looks like an e-mail address, UNC, or URL to a hyperlink. Word doesn't check the validity of the hyperlink.

Example

This example enables Word to automatically replace any Internet or network paths with hyperlinks when the paths are typed.

```
Options.AutoFormatAsYouTypeReplaceHyperlinks = True
```

This example returns the status of the **Internet and network paths with hyperlinks** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeReplaceHyperlinks
```



AutoFormatAsYouTypeReplaceOrdinalProperty

True if the ordinal number suffixes "st", "nd", "rd", and "th" are replaced with the same letters in superscript as you type. For example, "1st" is replaced with "1" followed by "st" formatted as superscript. Read/write **Boolean**.

Example

This example turns on the automatic replacement of ordinals with superscript letters.

```
Options.AutoFormatAsYouTypeReplaceOrdinals = True
```

This example returns the status of the **Ordinals (1st) with superscript** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeReplaceOrdinals
```



AutoFormatAsYouTypeReplacePlainT Property

True if manual emphasis characters are automatically replaced with character formatting as you type. For example, "***bold***" is changed to "**bold**" and "underline" is changed to "underline." Read/write **Boolean**.

Example

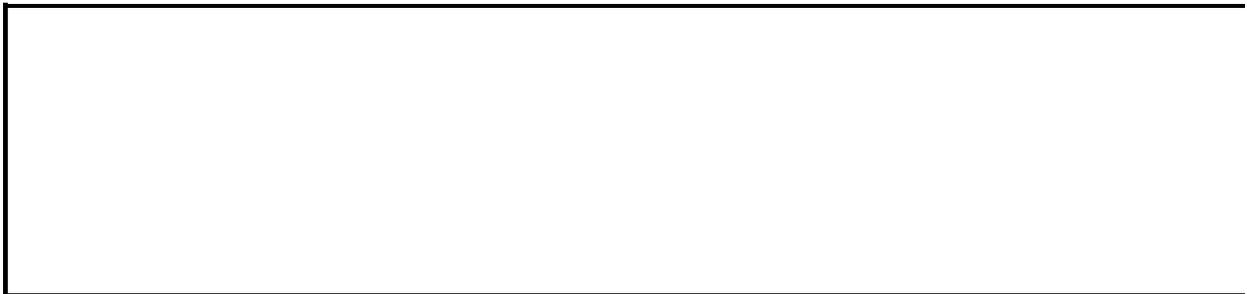
This example turns on the replacement of manual emphasis characters with character formatting.

```
Options.AutoFormatAsYouTypeReplacePlainTextEmphasis = True
```

This example returns the status of the ***Bold*** and **underline** with **real formatting** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = _  
    Options.AutoFormatAsYouTypeReplacePlainTextEmphasis
```



AutoFormatAsYouTypeReplaceQuoteProperty

True if straight quotation marks are automatically changed to smart (curly) quotation marks as you type. Read/write **Boolean**.

Example

This example turns on the automatic replacement of straight quotation marks with smart (curly) quotation marks as you type.

```
Options.AutoFormatAsYouTypeReplaceQuotes = True
```

This example returns the status of the **Straight quotes with smart quotes** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatReplaceQuotes
```



AutoFormatAsYouTypeReplaceSymbolProperty

True if two consecutive hyphens (--) are replaced with an en dash (–) or an em dash (—) as you type. Read/write **Boolean**.

Note If the hyphens are typed with leading and trailing spaces, Word replaces the hyphens with an en dash; if there are no trailing spaces, the hyphens are replaced with an em dash.

Example

This example turns on the replacement of hyphens with symbols as you type.

```
Options.AutoFormatAsYouTypeReplaceSymbols = True
```

This example returns the status of the **Symbol characters (--)** with **symbols (—)** option on the **AutoFormat As You Type** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatAsYouTypeReplaceSymbols
```



AutoFormatDeleteAutoSpaces Property

True if spaces inserted between Japanese and Latin text will be deleted when Microsoft Word formats a document or range automatically. Read/write **Boolean**.

expression.AutoFormatDeleteAutoSpaces

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to automatically delete spaces between Japanese and Latin text, and then it formats the current selection.

```
Options.AutoFormatDeleteAutoSpaces = True  
Selection.Range.AutoFormat
```



AutoFormatMatchParentheses Property

True if improperly paired parentheses are corrected when Microsoft Word formats a document or range automatically. Read/write **Boolean**.

expression.**AutoFormatMatchParentheses**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to automatically correct pairs of parentheses, and then it formats the current selection.

```
Options.AutoFormatMatchParentheses = True  
Selection.Range.AutoFormat
```



AutoFormatPlainTextWordMail Property

-

True if Word automatically formats plain-text e-mail messages when you open them in Word. Read/write **Boolean**.

Example

This example sets Word to automatically format any plain-text e-mail messages that are opened.

```
Options.AutoFormatPlainTextWordMail = True
```

This example returns the status of the **Plain text WordMail documents** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatPlainTextWordMail
```



AutoFormatPreserveStyles Property

-
True if previously applied styles are preserved when Word formats a document or range automatically. Read/write **Boolean**.

Example

This example sets Word to preserve existing styles and to format headings, lists, and other paragraphs with styles when formatting automatically. Word then formats the current selection automatically.

With Options

```
.AutoFormatPreserveStyles = True  
.AutoFormatApplyHeadings = True  
.AutoFormatApplyLists = True  
.AutoFormatApplyOtherParas = True
```

End With

```
Selection.Range.AutoFormat
```

This example returns the status of the **Styles** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatPreserveStyles
```



AutoFormatReplaceFarEastDashes Property

True if long vowel sound and dash use is corrected when Microsoft Word formats a document or range automatically. Read/write **Boolean**.

expression.**AutoFormatReplaceFarEastDashes**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to automatically correct the use of long vowel sounds and dashes, and then it formats the current selection.

```
Options.AutoFormatReplaceFarEastDashes = True  
Selection.Range.AutoFormat
```



AutoFormatReplaceFractions Property

True if typed fractions are replaced with fractions from the current character set when Word formats a document or range automatically. For example, "1/2" is replaced with "½." Read/write **Boolean**.

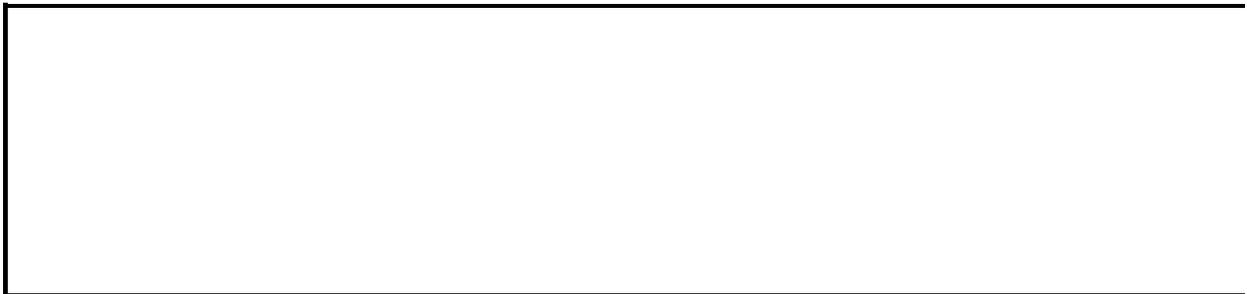
Example

This example turns on the replacement of typed fractions, and then it formats the current selection automatically.

```
Options.AutoFormatReplaceFractions = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Fractions (1/2) with fraction character (½)** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean  
  
blnAutoFormat = Options.AutoFormatReplaceFractions
```



AutoFormatReplaceHyperlinks Property

-

True if e-mail addresses, server and share names (also known as UNC paths), and Internet addresses (also known as URLs) are automatically formatted whenever Word AutoFormats a document or range. Read/write **Boolean**.

Remarks

Word changes any text that looks like an e-mail address, UNC, or URL to a hyperlink. Word doesn't check the validity of the hyperlink.

Example

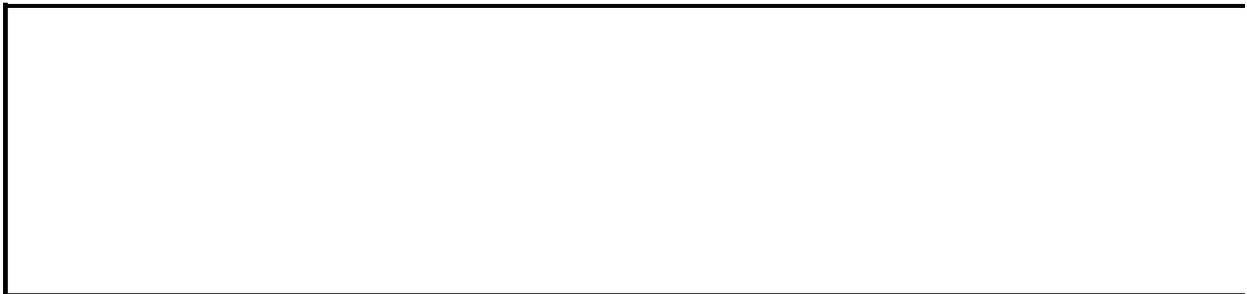
This example enables replacement of any Internet or network paths with hyperlinks, and then it formats the selection automatically.

```
Options.AutoFormatReplaceHyperlinks = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Internet and network paths with hyperlinks** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatReplaceHyperlinks
```



AutoFormatReplaceOrdinals Property

True if the ordinal number suffixes "st", "nd", "rd", and "th" are replaced with the same letters in superscript when Word formats a document or range automatically. For example, "1st" is replaced with "1" followed by "st" formatted as superscript. Read/write **Boolean**.

Example

This example turns on the automatic replacement of ordinals with superscript, and then it formats the current selection automatically.

```
Options.AutoFormatReplaceOrdinals = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Ordinals (1st) with superscript** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatReplaceOrdinals
```



AutoFormatReplacePlainTextEmphas Property

True if manual emphasis characters are replaced with character formatting when Word formats a document or range automatically. For example, "***bold***" is changed to "**bold**" and "underline" is changed to "underline." Read/write **Boolean**.

Example

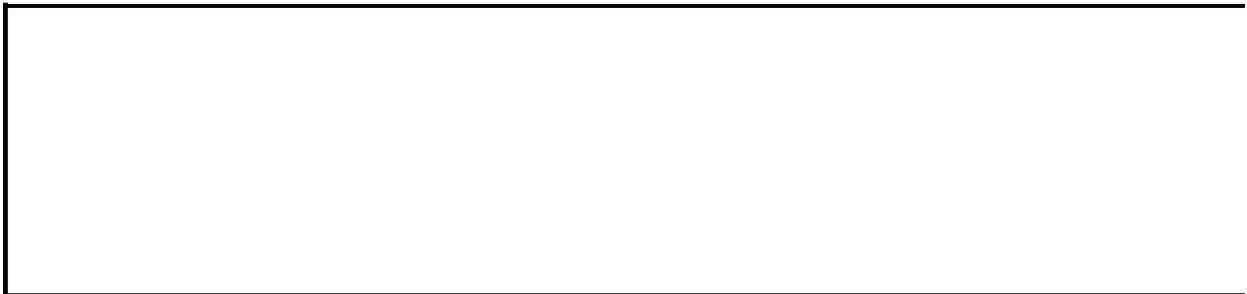
This example turns on the replacement of manual emphasis characters with character formatting

```
Options.AutoFormatReplacePlainTextEmphasis = True  
Selection.Range.AutoFormat
```

This example returns the status of the ***Bold*** and **underline with real formatting** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean
```

```
blnAutoFormat = Options.AutoFormatReplacePlainTextEmphasis
```



AutoFormatReplaceQuotes Property

-

True if straight quotation marks are automatically changed to smart (curly) quotation marks when Word formats a document or range automatically.
Read/write **Boolean**.

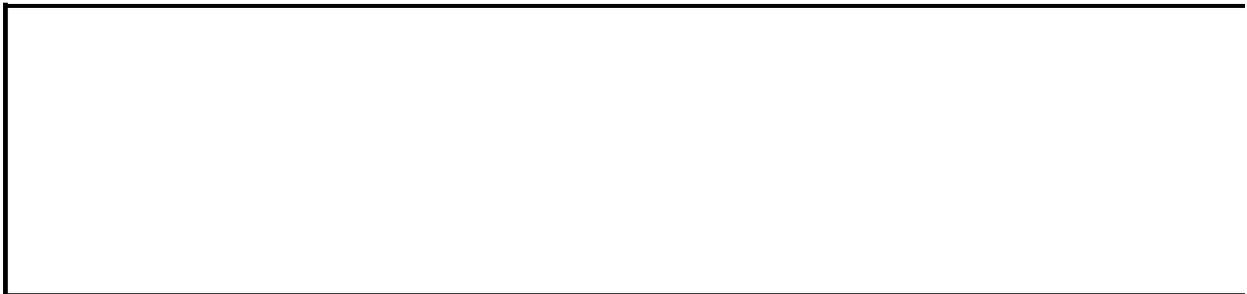
Example

This example turns on the automatic replacement of straight quotation marks with smart (curly) quotation marks, and then it formats the current selection automatically.

```
Options.AutoFormatReplaceQuotes = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Straight quotes with smart quotes** option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean  
  
blnAutoFormat = Options.AutoFormatReplaceQuotes
```



AutoFormatReplaceSymbols Property

-

True if two consecutive hyphens (--) are replaced by an en dash (–) or an em dash (—) when Word formats a document or range automatically. Read/write **Boolean**.

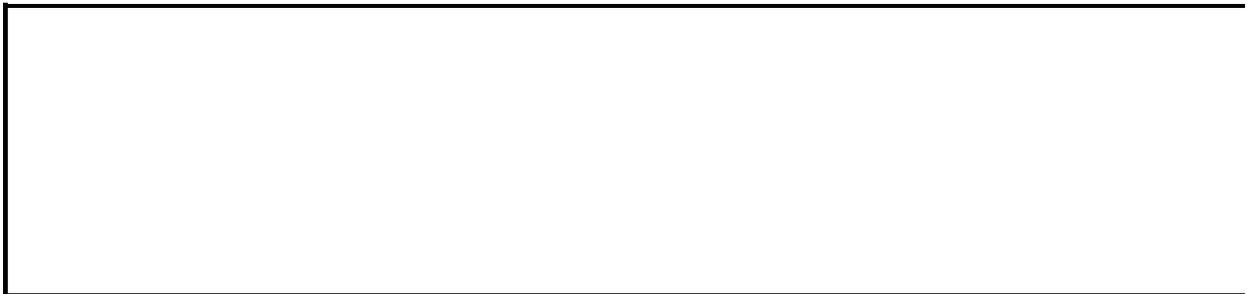
Example

This example turns on the replacement of hyphens with symbols, and then it formats the current selection automatically.

```
Options.AutoFormatReplaceSymbols = True  
Selection.Range.AutoFormat
```

This example returns the status of the **Symbol characters (--)** with symbols (—) option on the **AutoFormat** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnAutoFormat as Boolean  
  
blnAutoFormat = Options.AutoFormatReplaceSymbols
```



AutoFormatType Property

Returns the type of automatic formatting that's been applied to the specified table. Read-only **Long**. Can be one of the following [WdTableFormat](#) constants.

WdTableFormat can be one of these WdTableFormat constants.

wdTableFormat3DEffects1

wdTableFormat3DEffects2

wdTableFormat3DEffects3

wdTableFormatClassic1

wdTableFormatClassic2

wdTableFormatClassic3

wdTableFormatClassic4

wdTableFormatColorful1

wdTableFormatColorful2

wdTableFormatColorful3

wdTableFormatColumns1

wdTableFormatColumns2

wdTableFormatColumns3

wdTableFormatColumns4

wdTableFormatColumns5

wdTableFormatContemporary

wdTableFormatElegant

wdTableFormatGrid1

wdTableFormatGrid2

wdTableFormatGrid3

wdTableFormatGrid4

wdTableFormatGrid5

wdTableFormatGrid6

wdTableFormatGrid7

wdTableFormatGrid8
wdTableFormatList1
wdTableFormatList2
wdTableFormatList3
wdTableFormatList4
wdTableFormatList5
wdTableFormatList6
wdTableFormatList7
wdTableFormatList8
wdTableFormatNone
wdTableFormatProfessional
wdTableFormatSimple1
wdTableFormatSimple2
wdTableFormatSimple3
wdTableFormatSubtle1
wdTableFormatSubtle2
wdTableFormatWeb1
wdTableFormatWeb2
wdTableFormatWeb3

Note Use the [AutoFormat](#) method to apply automatic formatting to a table.

Example

This example formats the first table in the active document to use the Classic 1 AutoFormat if the current format is Simple 1, Simple 2, or Simple 3.

```
If ActiveDocument.Tables.Count >= 1 Then
    If ActiveDocument.Tables(1).AutoFormatType <= wdTableFormatSimple1
        ActiveDocument.Tables(1).AutoFormat _
            Format:=wdTableFormatClassic1
    End If
End If
```



AutoHyphenation Property

-

True if automatic hyphenation is turned on for the specified document.
Read/write **Boolean**.

Example

This example turns on automatic hyphenation, with a hyphenation zone of 0.25 inch. Words in all capital letters aren't hyphenated.

```
With ActiveDocument
    .HyphenationZone = InchesToPoints(0.25)
    .HyphenateCaps = False
    .AutoHyphenation = True
End With
```



AutoInsert Property

-
True if a caption is automatically added when the item is inserted into a document. Read/write **Boolean**.

Example

This example enables Word to add captions to tables automatically. Then the example collapses the selection to an insertion point, and inserts a table. A caption is automatically added to the new table.

```
AutoCaptions("Microsoft Word Table").AutoInsert = True  
Selection.Collapse Direction:=wdCollapseStart  
ActiveDocument.Tables.Add Range:=Selection.Range, _  
    NumRows:=2, NumColumns:=2
```


AutoKeyboardSwitching Property

True if Microsoft Word automatically switches the keyboard language to match what you're typing at any given time. Read/write **Boolean**.

expression.**AutoKeyboardSwitching**

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

Remarks

To use this property, you must have the [CheckLanguage](#) property set to **True**.

For more information on using Word with multiple languages, see [Troubleshoot Multilingual Text and Automatic Language Detection](#).

Example

This example asks the user to choose whether or not to enable automatic keyboard switching for multilingual documents.

```
x = MsgBox("Enable automatic keyboard switching?", vbYesNo)
If x = vbYes Then
    Application.CheckLanguage = True
    Options.AutoKeyboardSwitching = True
    MsgBox "Automatic keyboard switching enabled!"
End If
```



↳ [Show All](#)

AutoLayout Property

Returns or sets an [MsoTriState](#) constant that determines the automatic positioning of the nodes and connectors in a diagram. Read/write.

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse Disables automatic layout.

msoTriStateMixed Not used for this property.

msoTriStateToggle Not used for this property.

msoTrue Automatically positions nodes and connectors in a diagram.

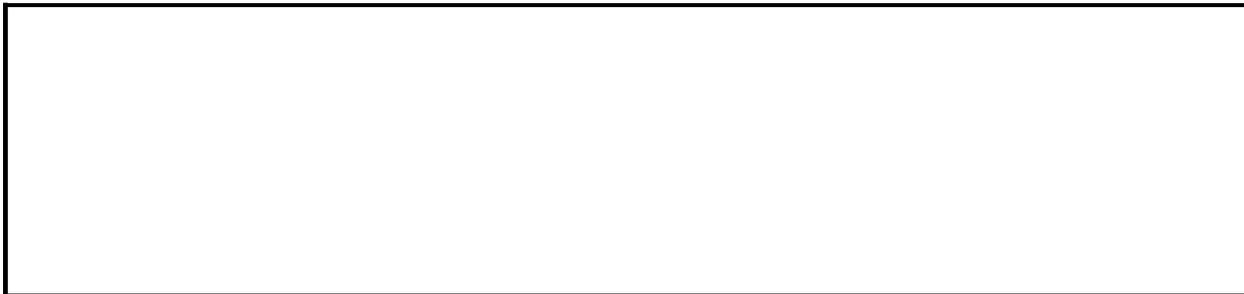
expression.**AutoLayout**

expression Required. An expression that returns a [Diagram](#) object.

Example

This example creates a diagram in the current document and automatically positions the nodes and connectors.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add a pyramid diagram to current document and first child node  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram( _  
        Type:=msoDiagramPyramid, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child node  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    'Enable automatic positioning of the diagram nodes  
    'and convert diagram to a radial diagram  
    With dgnNode.Diagram  
        .AutoLayout = msoTrue  
        .Convert Type:=msoDiagramRadial  
    End With  
  
End Sub
```



↳ [Show All](#)

AutoLength Property

-
MsoTrue to automatically sets the length of the callout line. Read-only
[MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse To set the length of the callout line manually.

msoTriStateMixed Not used with this property.

msoTriStateToggle Not used with this property.

msoTrue To automatically set the length of the callout line.

expression.**AutoLength**

expression Required. An expression that returns a [CalloutFormat](#) object.

Remarks

Use the [AutomaticLength](#) method to set this property to **msoTrue**, and use the [CustomLength](#) method to set this property to **msoFalse**.

Example

This example creates a new document and adds a callout to the new document, and then sets the length of the callout manually.

```
Sub AutoCalloutLength()  
    Dim docNew As Document  
    Dim shpCallout As Shape  
    Set docNew = Documents.Add  
    Set shpCallout = docNew.Shapes.AddCallout(Type:=msoCalloutFour,  
        Left:=15, Top:=15, Width:=150, Height:=200)  
    With shpCallout.Callout  
        If .AutoLength = msoTrue then  
            .CustomLength 50  
        End If  
    End With  
End Sub
```



Autoload Property

-

True if the specified add-in is automatically loaded when Word is started. Add-ins located in the Startup folder in the Word program folder are automatically loaded. Read-only **Boolean**.

Example

This example displays the name of each add-in that is automatically loaded when Word is started.

```
Dim addinLoop as AddIn
Dim blnFound as Boolean

blnFound = False

For Each addinLoop In AddIns
    With addinLoop
        If .Autoload = True Then
            MsgBox .Name
            blnFound = True
        End If
    End With
Next addinLoop

If blnFound <> True Then _
    MsgBox "No add-ins were loaded automatically."
```

This example determines whether the add-in named "Gallery.dot" was automatically loaded.

```
Dim addinLoop as AddIn

For Each addinLoop In AddIns
    If InStr(LCase$(addinLoop.Name), "gallery.dot") > 0 Then
        If addinLoop.Autoload = True Then MsgBox "Autoload"
    End If
Next addinLoop
```



AutomaticallyUpdate Property

-

True if the style is automatically redefined based on the selection. **False** if Word prompts for confirmation before redefining the style based on the selection. A style can be redefined when it's applied to a selection that has the same style but different manual formatting. Read/write **Boolean**.

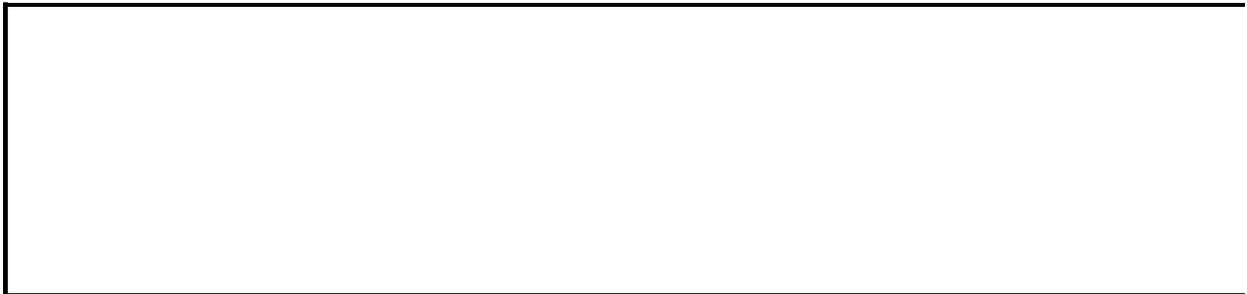
Example

This example creates a style named "Style1" that can be redefined without the need for confirmation.

```
Dim docNew as Document
Dim styleNew as Style

Set docNew = Documents.Add
Set styleNew = docNew.Styles.Add("Style1")

With styleNew
    .BaseStyle = docNew.Styles(wdStyleNormal)
    .ParagraphFormat.LineSpacingRule = wdLineSpaceDouble
    .AutomaticallyUpdate = True
End With
```



↳ [Show All](#)

AutomationSecurity Property

Returns or sets an [MsoAutomationSecurity](#) constant that represents the security mode Microsoft Word uses when programmatically opening files. Read/write.

MsoAutomationSecurity can be one of these MsoAutomationSecurity constants.

msoAutomationSecurityByUI Uses the security setting specified in the **Security** dialog box.

msoAutomationSecurityForceDisable Disables all macros in all files opened programmatically without showing any security alerts.

msoAutomationSecurityLow Enables all macros. This is the default value when the application is started.

expression.**AutomationSecurity**

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

Remarks

This property is automatically set to **msoAutomationSecurityLow** when Word is started. Therefore, to avoid breaking solutions that rely on the default setting, you should be careful to reset this property to **msoAutomationSecurityLow** after programmatically opening a file. Also, this property should be set immediately before and after opening a file programmatically to avoid malicious subversion.

Setting **ScreenUpdating** to **False** does not affect alerts and will not affect security warnings. The **DisplayAlerts** setting will not apply to security warnings. For example, if the user sets **DisplayAlerts** equal to **False** and **AutomationSecurity** to **msoAutomationSecurityByUI**, while the user is on Medium security level, then there will be security warnings while the macro is running. This allows the macro to trap file open errors, while still showing the security warning if the file open succeeds.

Example

This example captures the current automation security setting, changes the setting to disable macros, displays the **Open** dialog box, and after opening the selected document, sets the automation security back to its original setting.

```
Sub Security()  
    Dim secAutomation As MsoAutomationSecurity  
  
    With Application  
        secAutomation = .AutomationSecurity  
        .AutomationSecurity = msoAutomationSecurityForceDisable  
        With .FileDialog(msoFileDialogOpen)  
            .Show  
            .Execute  
        End With  
        .AutomationSecurity = secAutomation  
    End With  
  
End Sub
```



↳ [Show All](#)

AutoShapeType Property

Returns or sets the shape type for the specified [Shape](#) or [ShapeRange](#) object, which must represent an AutoShape other than a line or freeform drawing. Read/write [MsoAutoShapeType](#).

MsoAutoShapeType can be one of these MsoAutoShapeType constants.

msoShape24pointStar

msoShape4pointStar

msoShape8pointStar

msoShapeActionButtonBeginning

msoShapeActionButtonDocument

msoShapeActionButtonForwardorNext

msoShapeActionButtonHome

msoShapeActionButtonMovie

msoShapeActionButtonSound

msoShapeBalloon

msoShapeBentUpArrow

msoShapeBlockArc

msoShapeChevron

msoShapeCloudCallout

msoShapeCube

msoShapeCurvedDownRibbon

msoShapeCurvedRightArrow

msoShapeCurvedUpRibbon

msoShapeDonut

msoShapeDoubleBracket

msoShapeDownArrow

msoShapeDownRibbon

msoShapeExplosion2

msoShapeFlowchartCard
msoShapeFlowchartConnector
msoShapeFlowchartDecision
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
msoShapeMoon
msoShapeNoSymbol
msoShapeNotchedRightArrow
msoShapeNotPrimitive
msoShapeOctagon
msoShapeOval
msoShapeOvalCallout
msoShapeParallelogram
msoShapePentagon
msoShapePlaque
msoShapeQuadArrowCallout

msoShapeRectangularCallout
msoShapeRightArrow
msoShapeRightBrace
msoShapeRightTriangle
msoShapeRoundedRectangularCallout
msoShapeStripedRightArrow
msoShapeTrapezoid
msoShapeUpArrowCallout
msoShapeUpDownArrowCallout
msoShapeUTurnArrow
msoShapeWave
msoShape16pointStar
msoShape32pointStar
msoShape5pointStar
msoShapeActionButtonBackorPrevious
msoShapeActionButtonCustom
msoShapeActionButtonEnd
msoShapeActionButtonHelp
msoShapeActionButtonInformation
msoShapeActionButtonReturn
msoShapeArc
msoShapeBentArrow
msoShapeBevel
msoShapeCan
msoShapeCircularArrow
msoShapeCross
msoShapeCurvedDownArrow
msoShapeCurvedLeftArrow
msoShapeCurvedUpArrow
msoShapeDiamond
msoShapeDoubleBrace
msoShapeDoubleWave
msoShapeDownArrowCallout

msoShapeExplosion1
msoShapeFlowchartAlternateProcess
msoShapeFlowchartCollate
msoShapeFlowchartData
msoShapeFlowchartDelay
msoShapeQuadArrow
msoShapeRectangle
msoShapeRegularPentagon
msoShapeRightArrowCallout
msoShapeRightBracket
msoShapeRoundedRectangle
msoShapeSmileyFace
msoShapeSun
msoShapeUpArrow
msoShapeUpDownArrow
msoShapeUpRibbon
msoShapeVerticalScroll

expression.**AutoShapeType**

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

Remarks

When you change the type of a shape, the shape retains its size, color, and other attributes.

Example

This example replaces all 16-point stars with 32-point stars in the active document.

```
Sub ReplaceAutoShape()  
    Dim docNew As Document  
    Dim shpStar As Shape  
    Set docNew = ActiveDocument  
    For Each shpStar In docNew.Shapes  
        If shpStar.AutoShapeType = msoShape16pointStar Then  
            shpStar.AutoShapeType = msoShape32pointStar  
        End If  
    Next  
End Sub
```



↳ [Show All](#)

AutoSize Property

▶ [AutoSize property as it applies to the **CheckBox** object.](#)

True sizes the check box or text frame according to the font size of the surrounding text. **False** sizes the check box or text frame according to the [Size](#) property. Read/write **Boolean**.

expression.**AutoSize**

expression Required. An expression that returns a [CheckBox](#) object.

▶ [AutoSize property as it applies to the **TextFrame** object.](#)

Returns or sets a **Long** that represents whether a text frame is sized automatically. Read/write.

expression.**AutoSize**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example sets the size of the check box named "Check1" to Auto and then selects the check box.

```
With ActiveDocument.FormFields("Check1").CheckBox  
    .AutoSize = True  
    .Value = True  
End With
```



AutoTextEntries Property

Returns an [AutoTextEntries](#) collection that represents all the AutoText entries in the specified template. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example deletes the AutoText entry named "Hello" if the entry exists in the attached template.

```
Dim atEntry As AutoTextEntry

For Each atEntry In _
    ActiveDocument.AttachedTemplate.AutoTextEntries
    If atEntry.Name = "asdf" Then atEntry.Delete
    Debug.Print atEntry.Name
Next atEntry
```

This example adds an AutoText entry named "Temp" to the Normal template. The contents of the AutoText entry (the first word in the document) are then displayed in a message box.

```
Dim atEntry As AutoTextEntry

Set atEntry = _
    NormalTemplate.AutoTextEntries.Add(Name:="Temp", _
    Range:=ActiveDocument.Words(1))

MsgBox atEntry.Value
```

This example stores the contents of the selection as an AutoText entry named "Address" in the attached template.

```
If Len(Selection.Text) > 1 Then
    ActiveDocument.AttachedTemplate.AutoTextEntries.Add _
        Range:=Selection.Range, Name:="Address"
End If
```



AutoUpdate Property

-
True if the specified link is updated automatically when the container file is opened or when the source file is changed. Read/write **Boolean**.

Example

This example updates any shapes in the active document that are linked OLE objects if Word isn't set to update links automatically.

```
Dim shapeLoop as Shape

For Each shapeLoop In ActiveDocument.Shapes
    With shapeLoop
        If .Type = msoLinkedOLEObject Then
            If .LinkFormat.AutoUpdate = False Then
                .LinkFormat.Update
            End If
        End If
    End With
Next s
```

This example updates any fields in the active document that aren't updated automatically.

```
Dim fieldLoop as Field

For Each fieldLoop In ActiveDocument.Fields
    If fieldLoop.LinkFormat.AutoUpdate = False Then _
        fieldLoop.LinkFormat.Update
Next fieldLoop
```



AutoVersion Property

Returns or sets the state of the option for automatically saving document versions. Can be one of the following read/write [WdAutoVersions](#) constants.

WdAutoVersions can be one of these WdAutoVersions constants.

wdAutoVersionOff

wdAutoVersionOnClose

expression.**AutoVersion**

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

Note When the **AutoVersion** property is set to **wdAutoVersionOnClose**, a document version is automatically saved when the document is closed.

Example

This example disables the option to save a document version automatically when the active document is closed.

```
ActiveDocument.Versions.AutoVersion = wdAutoVersionOff
```

This example displays a message in the status bar if the option to save a document version automatically is active for Report.doc.

```
If Documents("Report.doc").Versions.AutoVersion = _  
    wdAutoVersionOnClose Then  
    StatusBar = "A version will be automatically saved"  
End If
```



AutoWordSelection Property

-

True if dragging selects one word at a time instead of one character at a time.
Read/write **Boolean**.

Example

This example sets Word to select individual characters instead of entire words when you select by dragging.

```
Options.AutoWordSelection = False
```

This example returns the status of the **When selecting, automatically select entire word** option on the **Edit** tab in the **Options** dialog box.

```
Dim blnAutoSelect as Boolean
```

```
blnAutoSelect = Options.AutoWordSelection
```



BackColor Property

-
Returns or sets a [ColorFormat](#) object that represents the background color for the specified fill or patterned line. Read/write.

Example

This example adds a rectangle to the active document and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
Dim docActive As Document

Set docActive = ActiveDocument

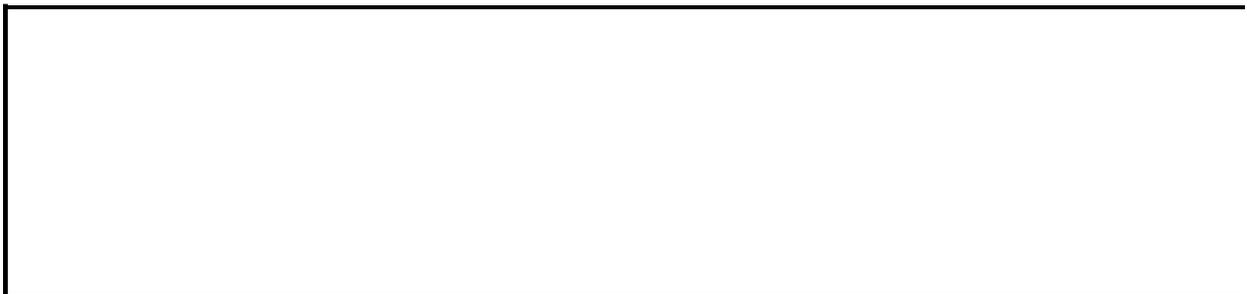
With docActive.Shapes.AddShape(msoShapeRectangle, _
    90, 90, 90, 50).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(170, 170, 170)
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

This example adds a patterned line to the active document.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes.AddLine(10, 100, 250, 0).Line
    .Weight = 6
    .ForeColor.RGB = RGB(0, 0, 255)
    .BackColor.RGB = RGB(128, 0, 0)
    .Pattern = msoPatternDarkDownwardDiagonal
End With
```



Background Property

-

Returns a **Shape** object that represents the background image for the specified document. Read-only.

Note Backgrounds are visible only in web layout view.

Example

This example sets the background color for web layout view to light gray for the active window.

```
ActiveDocument.ActiveWindow.View.Type = wdWebView  
With ActiveDocument.Background.Fill  
    .Visible = True  
    .ForeColor.RGB = RGB(192, 192, 192)  
End With
```

This example sets the background bitmap image of web layout view to Bubbles.bmp.

```
ActiveDocument.ActiveWindow.View.Type = wdWebView  
ActiveDocument.Background.Fill.UserPicture _  
    PictureFile:="C:\Windows\Bubbles.bmp"
```



BackgroundOpen Property

True for Microsoft Word to open Web documents in the background. Read/write **Boolean**.

expression.**BackgroundOpen**

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

Remarks

While Microsoft Word is opening a large Web document in the background, users can continue to type and choose commands in another document. However, until the Web document is fully opened, Word Visual Basic for Applications functions are disabled for the document being opened.

Example

This example toggles between opening large Web documents in the background and not opening them in the background.

```
Sub BackOpen()  
    If Options.BackgroundOpen = False Then  
        Options.BackgroundOpen = True  
    Else  
        Options.BackgroundOpen = False  
    End If  
End Sub
```



↳ [Show All](#)

BackgroundPatternColor Property

Returns or sets the 24-bit color that's applied to the background of the [Shading](#) object. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**BackgroundPatternColor**

expression Required. An expression that returns a [Shading](#) object.

Example

This example applies turquoise background shading to the first paragraph in the active document.

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.Shading.BackgroundPatternColor = _
    wdColorTurquoise
```

This example adds a table at the insertion point and then applies light gray background shading to the first cell.

```
Selection.Collapse Direction:=wdCollapseStart
Set myTable = _
    ActiveDocument.Tables.Add(Range:=Selection.Range, _
        NumRows:=2, NumColumns:=2)
myTable.Cell(1, 1).Shading.BackgroundPatternColor = _
    wdColorGray25
```



BackgroundPatternColorIndex Property

Returns or sets the color that's applied to the background of the **Shading** object.
Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**BackgroundPatternColorIndex**

expression Required. An expression that returns one of the objects in the

Applies To list.

Example

This example applies cyan background shading to the first paragraph in the active document.

```
Dim rngTemp As Range
```

```
Set rngTemp = ActiveDocument.Paragraphs(1).Range  
rngTemp.Shading.BackgroundPatternColorIndex = wdTurquoise
```

This example adds a table at the insertion point and then applies light gray background shading to the first cell.

```
Dim tableNew As Table
```

```
Selection.Collapse Direction:=wdCollapseStart  
Set tableNew = ActiveDocument.Tables.Add(Range:=Selection.Range, _  
    NumRows:=2, NumColumns:=2)  
tableNew.Cell(1, 1).Shading.BackgroundPatternColorIndex = _  
    wdGray25
```


BackgroundPrintingStatus Property

-

Returns the number of print jobs in the background printing queue. Read-only
Long.

Example

This example returns the number of Word print jobs currently queued up for background printing.

```
Dim lngStatus As Long
```

```
If Options.PrintBackground = True Then  
    lngStatus = Application.BackgroundPrintingStatus  
End If
```

If the number of print jobs is greater than 0 (zero), this example displays a message in the status bar.

```
If Application.BackgroundPrintingStatus > 0 Then  
    StatusBar = Application.BackgroundPrintingStatus _  
        & " print jobs are queued up"  
End If
```



BackgroundSave Property

-

True if Word saves documents in the background. When Word is saving in the background, users can continue to type and to choose commands. Read/write **Boolean**.

Example

This example allows users to continue working in a document while Word is saving it.

```
Options.BackgroundSave = True
```

This example returns the current status of the **Allow background saves** option on the **Save** tab in the **Options** dialog box.

```
Dim blnAutoSave As Boolean
```

```
blnAutoSave = Options.BackgroundSave
```



BackgroundSavingStatus Property

-
Returns the number of files queued up to be saved in the background. Read-only
Long.

Example

This example displays in the status bar the number of documents currently being saved.

```
Options.BackgroundSave =True
Documents.Add
ActiveDocument.SaveAs
    While Application.BackgroundSavingStatus <> 0
        StatusBar = "Documents remaining to save: " _
            & Application.BackgroundSavingStatus
    DoEvents
Wend
```



BaseStyle Property

Returns or sets an existing style on which you can base the formatting of another style. To set this property, specify either the local name of the base style, an integer or a **WdBuiltinStyle** constant, or an object that represents the base style. Read/write **Variant**.

For a list of the **WdBuiltinStyle** constants, see the [Style](#) property.

Example

This example creates a new document and then adds a new paragraph style named "myHeading." It assigns Heading 1 as the base style for the new style . A left indent of 1 inch (72 points) is then specified for the new style.

```
Dim docNew As Document
Dim styleNew As Style

Set docNew = Documents.Add
Set styleNew = docNew.Styles.Add("NewHeading1")
With styleNew
    .BaseStyle = docNew.Styles(wdStyleHeading1)
    .ParagraphFormat.LeftIndent = 72
End With
```

This example returns the base style that's used for the Body Text paragraph style.

```
Dim styleBase As Style

styleBase = ActiveDocument.Styles(wdStyleBodyText).BaseStyle
MsgBox styleBase
```



BeginArrowheadLength Property

Returns or sets the length of the arrowhead at the beginning of the specified line. Read/write [MsoArrowheadLength](#).

MsoArrowheadLength can be one of these MsoArrowheadLength constants.

msoArrowheadLengthMixed

msoArrowheadShort

msoArrowheadLengthMedium

msoArrowheadLong

expression.**BeginArrowheadLength**

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

Example

This example adds a line to the active document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(100, 100, 200, 300).Line
```

```
    .BeginArrowheadLength = msoArrowheadShort
```

```
    .BeginArrowheadStyle = msoArrowheadOval
```

```
    .BeginArrowheadWidth = msoArrowheadNarrow
```

```
    .EndArrowheadLength = msoArrowheadLong
```

```
    .EndArrowheadStyle = msoArrowheadTriangle
```

```
    .EndArrowheadWidth = msoArrowheadWide
```

```
End With
```



BeginArrowheadStyle Property

Returns or sets the style of the arrowhead at the beginning of the specified line.
Read/write [MsoArrowheadStyle](#).

MsoArrowheadStyle can be one of these MsoArrowheadStyle constants.

msoArrowheadNone

msoArrowheadOval

msoArrowheadStyleMixed

msoArrowheadDiamond

msoArrowheadOpen

msoArrowheadStealth

msoArrowheadTriangle

expression.**BeginArrowheadStyle**

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

Example

This example adds a line to the active document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(100, 100, 200, 300).Line  
    .BeginArrowheadLength = msoArrowheadShort  
    .BeginArrowheadStyle = msoArrowheadOval  
    .BeginArrowheadWidth = msoArrowheadNarrow  
    .EndArrowheadLength = msoArrowheadLong  
    .EndArrowheadStyle = msoArrowheadTriangle  
    .EndArrowheadWidth = msoArrowheadWide  
End With
```



BeginArrowheadWidth Property

Returns or sets the width of the arrowhead at the beginning of the specified line. Read/write [MsoArrowheadWidth](#).

MsoArrowheadWidth can be one of these MsoArrowheadWidth constants.

msoArrowheadNarrow

msoArrowheadWidthMedium

msoArrowheadWide

msoArrowheadWidthMixed

expression.**BeginArrowheadWidth**

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

Example

This example adds a line to the first document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docFirst As Document
```

```
Set docFirst = Documents(1)
```

```
With docFirst.Shapes.AddLine(100, 100, 200, 300).Line
```

```
    .BeginArrowheadLength = msoArrowheadShort
```

```
    .BeginArrowheadStyle = msoArrowheadOval
```

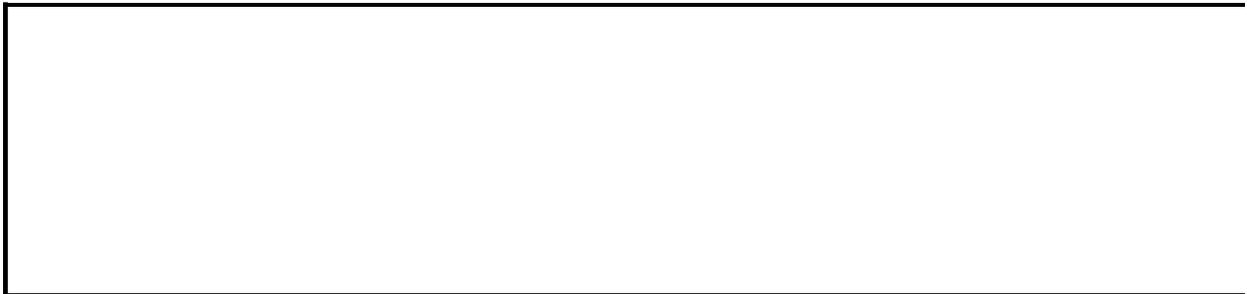
```
    .BeginArrowheadWidth = msoArrowheadNarrow
```

```
    .EndArrowheadLength = msoArrowheadLong
```

```
    .EndArrowheadStyle = msoArrowheadTriangle
```

```
    .EndArrowheadWidth = msoArrowheadWide
```

```
End With
```



Black Property

-
Sets or returns a **Long** that represents the black component of a CMYK color.
Read-only.

expression.**Black**

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

Example

This example creates a new shape, then retrieves the four CMYK values from an existing shape in the active document, and then sets the CMYK fill color of the new 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.Shapes(1)  
    Set shpStar = ActiveDocument.Shapes.AddShape _  
        (Type:=msoShape5pointStar, Left:=200, _  
         Top:=100, Width:=150, Height:=150)  
  
    'Get current shapes CMYK colors  
    With shpHeart.Fill.ForeColor  
        lngCyan = .Cyan  
        lngMagenta = .Magenta  
        lngYellow = .Yellow  
        lngBlack = .Black  
    End With  
  
    'Set new shape to current shapes CMYK colors  
    shpStar.Fill.ForeColor.SetCMYK _  
        Cyan:=lngCyan, Magenta:=lngMagenta, _  
        Yellow:=lngYellow, Black:=lngBlack  
End Sub
```



BlueScreen Property

-
True if Word displays text as white characters on a blue background. Read/write **Boolean**.

Example

This example asks users whether they want white text on a blue background and presents Yes and No buttons for their response.

```
If MsgBox("Do you want white on blue?", 36, _  
    "BlueScreen?") = vbYes Then  
    Options.BlueScreen = True  
Else  
    Options.BlueScreen = False  
End If
```



Bold Property

-
True if the font or range is formatted as bold. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

Example

This example formats the sixth word in a new document as bold.

```
Set newDoc = Documents.Add
Set myRange = newDoc.Content
myRange.InsertAfter "This is a test of bold."
myRange.Words(6).Bold = True
```

This example makes the entire selection bold if part of the selection is formatted as bold.

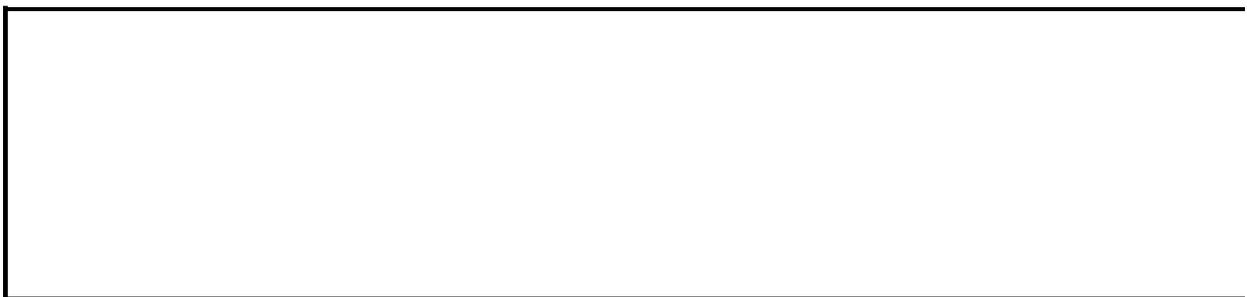
```
If Selection.Type = wdSelectionNormal Then
    If Selection.Font.Bold = wdUndefined Then _
        Selection.Font.Bold = True
Else
    MsgBox "You need to select some text."
End If
```

This example toggles the bold format for the selected text.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Range.Bold = wdToggle
End If
```

This example makes the first paragraph in the active document bold.

```
ActiveDocument.Paragraphs(1).Range.Bold = True
```



BoldBi Property

-
True if the font or range is formatted as bold. Returns **True**, **False** or **wdUndefined** (for a mixture of bold and non-bold text). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**BoldBi**

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

Remarks

The **BoldBi** property applies to text in a right-to-left language.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example makes the first paragraph in the active right-to-left language document bold.

```
ActiveDocument.Paragraphs(1).Range.BoldBi = True
```



BookFoldPrinting Property

True for Microsoft Word to print a document in a series of booklets so the printed pages can be folded and read as a book. Read/write **Boolean**.

expression.**BookFoldPrinting**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example turns the active document into a booklet that prints in four-page increments.

```
Sub Booklet()  
  With PageSetup  
    .BookFoldPrinting = True  
    .BookFoldPrintingSheets = 4  
  End With  
End Sub
```



BookFoldPrintingSheets Property

Returns or sets a **Long** which represents the number of pages for each booklet.
Read/write **Boolean**.

expression.**BookFoldPrintingSheets**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example turns the active document into a booklet that will print in sixteen-page booklets.

```
Sub Booklet()  
  With PageSetup  
    .BookFoldPrinting = True  
    .BookFoldPrintingSheets = 16  
  End With  
End Sub
```



↳ [Show All](#)

BookFoldRevPrinting Property

True for Microsoft Word to reverse the printing order for [book fold printing](#) of bidirectional or Asian language documents. Read/write **Boolean**.

expression.**BookFoldRevPrinting**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example switches from left-to-right book printing to right-to-left book printing for a bidirectional or Asian language document that will print in sixteen-page increments.

```
Sub BookletRev()  
  With PageSetup  
    .BookFoldRevPrinting = True  
    .BookFoldPrintingSheets = 16  
  End With  
End Sub
```



Bookmark Property

-

Returns or sets the name of the bookmark from which to collect table of authorities entries. Read/write **String**.

Remarks

The **Bookmark** property corresponds to the \b switch for a TOA (Table of Authorities) field.

Example

If a table of authorities exists in the active document, the entries are collected from the area defined by the bookmark named "area."

```
If ActiveDocument.TablesOfAuthorities.Count >= 1 Then
    ActiveDocument.TablesOfAuthorities(1).Bookmark = "area"
End If
```



BookmarkID Property

-

Returns the number of the bookmark that encloses the beginning of the specified selection or range; returns 0 (zero) if there's no corresponding bookmark. The number corresponds to the position of the bookmark in the document — 1 for the first bookmark, 2 for the second one, and so on. Read-only **Long**.

Example

This example displays the number of the bookmark that encloses the beginning of the selection.

```
MsgBox "Bookmark " & Selection.BookmarkID
```

This example adds a bookmark named "temp" at the beginning of the document if there's not already a bookmark set for that location.

```
Set myRange = ActiveDocument.Content  
myRange.Collapse Direction:=wdCollapseStart  
If myRange.BookmarkID = 0 Then  
    ActiveDocument.Bookmarks.Add Name:="temp", Range:=myRange  
End If
```



Bookmarks Property

Returns a [Bookmarks](#) collection that represents all the bookmarks in a document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example retrieves the starting and ending character positions for the first bookmark in the active document.

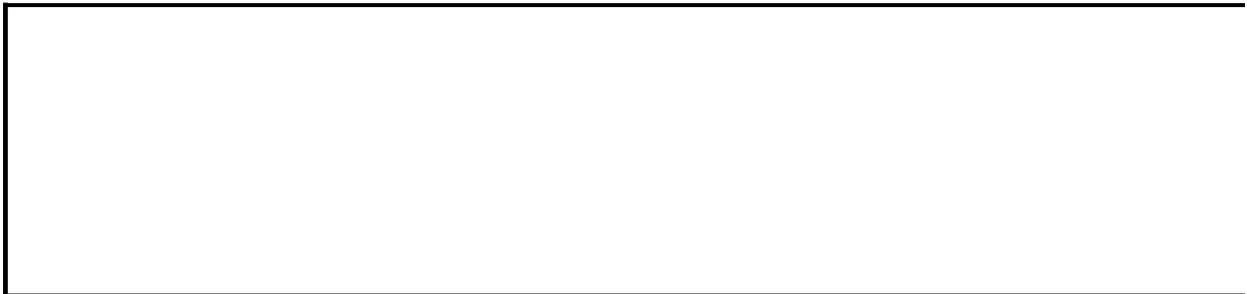
```
With ActiveDocument.Bookmarks(1)
    BookStart = .Start
    BookEnd = .End
End With
```

This example uses the `aMarks()` array to store the name of each bookmark contained in the active document.

```
If ActiveDocument.Bookmarks.Count >= 1 Then
    ReDim aMarks(ActiveDocument.Bookmarks.Count - 1)
    i = 0
    For Each aBookmark In ActiveDocument.Bookmarks
        aMarks(i) = aBookmark.Name
        i = i + 1
    Next aBookmark
End If
```

This example applies bold formatting to the first range of bookmarked text in the selection.

```
If Selection.Bookmarks.Count >= 1 Then
    Selection.Bookmarks(1).Range.Bold = True
End If
```



Border Property

Returns or sets whether the text in the specified callout is surrounded by a border. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

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.

Read/write **Long**.

Example

This example adds an oval to the active document 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.

```
Dim docActive As Document
Set docActive = ActiveDocument

With docActive.Shapes
    .AddShape msoShapeOval, 180, 200, 280, 130
    With .AddCallout(msoCalloutTwo, 420, 170, 170, 40)
        .TextFrame.TextRange.Text = "My oval"
        With .Callout
            .Accent = True
            .Border = False
        End With
    End With
End With
```



Borders Property

Returns a [Borders](#) collection that represents all the borders for the specified object.

expression.**Borders**

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 applies inside and outside borders to the first table in the active document.

```
Set myTable = ActiveDocument.Tables(1)
With myTable.Borders
    .InsideLineStyle = wdLineStyleSingle
    .OutsideLineStyle = wdLineStyleDouble
End With
```

This example applies a border around the first character in the selection. If nothing is selected, the border is applied to the first character after the insertion point.

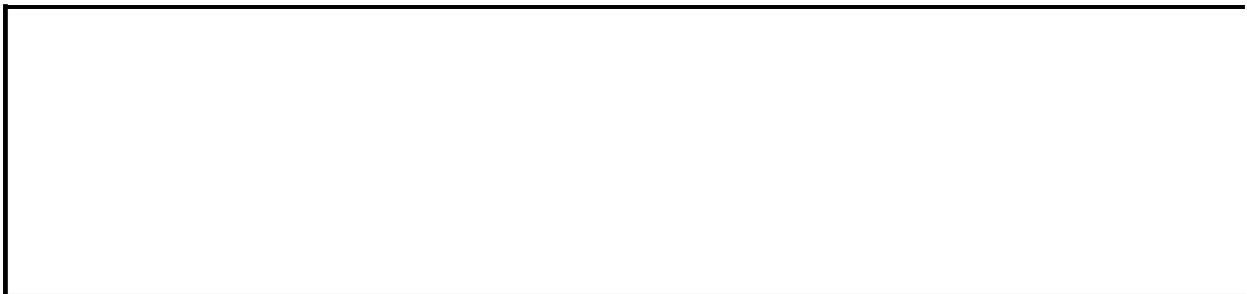
```
Selection.Characters(1).Borders.Enable = True
```

This example applies a bottom border below all centered paragraphs in the active document.

```
For Each para In ActiveDocument.Paragraphs
    If para.Alignment = wdAlignParagraphCenter Then
        para.Borders(wdBorderBottom).LineStyle = wdLineStyleSingle
        para.Borders(wdBorderBottom).LineWidth = wdLineWidth300pt
    End If
Next para
```

This example adds a border around all the pages in the current section.

```
For Each aBorder In Selection.Sections(1).Borders
    aBorder.ArtStyle = wdArtBasicBlackDots
    aBorder.ArtWidth = 6
Next aBorder
```



BottomMargin Property

-

Returns or sets the distance (in points) between the bottom edge of the page and the bottom boundary of the body text. Read/write **Single**.

Example

This example sets the bottom margin to 72 points (1 inch) and the top margin to 2 inches for the active document. The **InchesToPoints** method is used to convert inches to points.

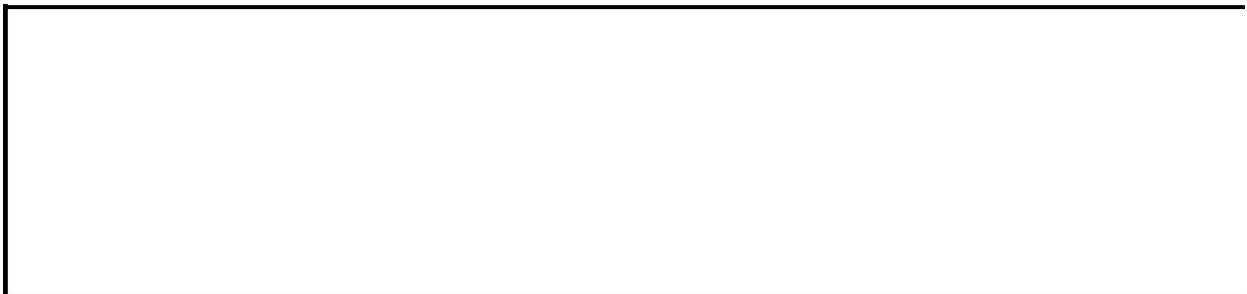
```
With ActiveDocument.PageSetup
    .BottomMargin = 72
    .TopMargin = InchesToPoints(2)
End With
```

This example sets the bottom margin to 2.5 inches for all the sections in the current selection.

```
Selection.PageSetup.BottomMargin = InchesToPoints(2.5)
```

This example returns the bottom margin for section 1 in the selection. The **PointsToInches** method is used to convert the result to inches.

```
Dim sngMargin As Single
sngMargin = Selection.Sections(1).PageSetup.BottomMargin
MsgBox PointsToInches(sngMargin) & " inches"
```



BottomPadding Property

Returns or sets the amount of space (in points) to add below the contents of a single cell or all the cells in a table. Read/write **Single**.

expression.**BottomPadding**

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

Remarks

The setting of the **BottomPadding** property for a single cell overrides the setting of the **BottomPadding** property for the entire table.

Example

This example sets the bottom padding for the first table in the active document to 40 pixels.

```
ActiveDocument.Tables(1).BottomPadding = _  
    PixelsToPoints(40, True)
```



Brightness Property

-

Returns or sets the brightness of the specified picture or OLE object. The value for this property must be a number from 0.0 (dimkest) to 1.0 (brightest).
Read/write **Single**.

Example

This example sets the brightness for the first shape on the active document. The first shape must be either a picture or an OLE object.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
docActive.Shapes(1).PictureFormat.Brightness = 0.3
```



BrowseExtraFileTypes Property

-

Set this property to "text/html" to allow hyperlinked HTML files to be opened in Microsoft Word (instead of the default Internet browser). Read/write **String**.

Example

This example allows hyperlinked HTML files to be opened in Word (instead of the default Internet browser).

```
Application.BrowseExtraFileTypes = "text/html"
```



Browser Property

-
Returns a [Browser](#) object that represents the **Select Browse Object** tool on the vertical scroll bar. Read-only.

Example

This example moves to the next footnote reference mark in the active document.

```
With Application.Browser  
    .Target = wdBrowseFootnote  
    .Next  
End With
```

This example moves to the next field in the active document. The text from the initial selection to the next field is formatted as bold.

```
Selection.ExtendMode = True  
With Application.Browser  
    .Target = wdBrowseField  
    .Next  
End With  
With Selection  
    .Font.Bold = True  
    .ExtendMode = False  
    .Collapse Direction:=wdCollapseEnd  
End With
```



↳ [Show All](#)

BrowserLevel Property

▸ [As it applies to the **DefaultWebOptions** object.](#)

Returns or sets a **WdBrowserLevel** that represents the level of the Web browser for which you want to target new Web pages created in Microsoft Word.
Read/write.

WdBrowserLevel can be one of these WdBrowserLevel constants.

wdBrowserLevelMicrosoftInternetExplorer6

wdBrowserLevelMicrosoftInternetExplorer5

wdBrowserLevelV4

expression.**BrowserLevel**

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

Remarks

After you set the **BrowserLevel** property on the [DefaultWebOptions](#) object, the **BrowserLevel** property of any new Web pages you create in Word will be the same as the global setting.

▶ [As it applies to the WebOptions object.](#)

Returns or sets [WdBrowserLevel](#) that represents the level of Web browser at which you want to target the specified Web page. This property is ignored if the [OptimizeForBrowser](#) property is set to **False**. Read/write.

WdBrowserLevel can be one of these WdBrowserLevel constants.

wdBrowserLevelMicrosoftInternetExplorer6

wdBrowserLevelMicrosoftInternetExplorer5

wdBrowserLevelV4

expression.**BrowserLevel**

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

Example

▶ [As it applies to the **DefaultWebOptions** object.](#)

This example sets Word to optimize new Web pages for Microsoft Internet Explorer 5 and creates a Web page based on this setting.

```
With Application.DefaultWebOptions
    .BrowserLevel = wdBrowserLevelMicrosoftInternetExplorer5
    .OptimizeForBrowser = True
End With
Documents.Add DocumentType:=wdNewWebPage
```

▶ [As it applies to the **WebOptions** object.](#)

This example creates a new Web page and optimizes it for Microsoft Internet Explorer 5.

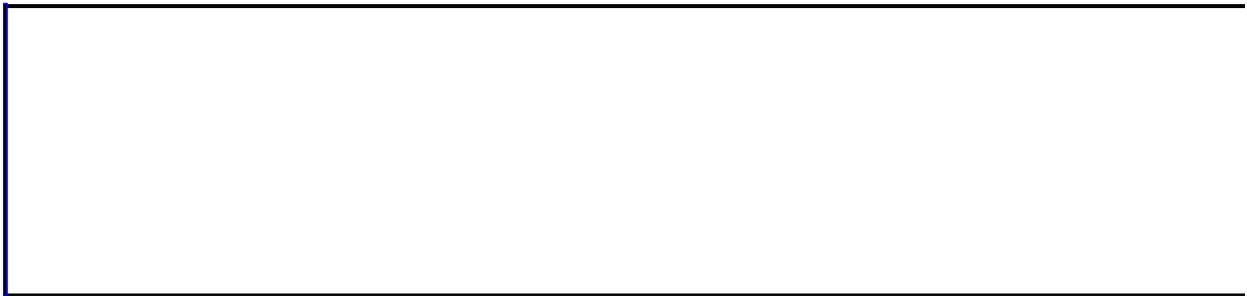
```
Documents.Add DocumentType:=wdNewWebPage
With ActiveDocument.WebOptions
    .BrowserLevel = wdBrowserLevelMicrosoftInternetExplorer5
    .OptimizeForBrowser = True
End With
```



BrowseWidth Property

Returns the width (in points) of the area in which text wraps in the specified pane. Read-only **Long**.

Note This property works only when you're in web layout view.



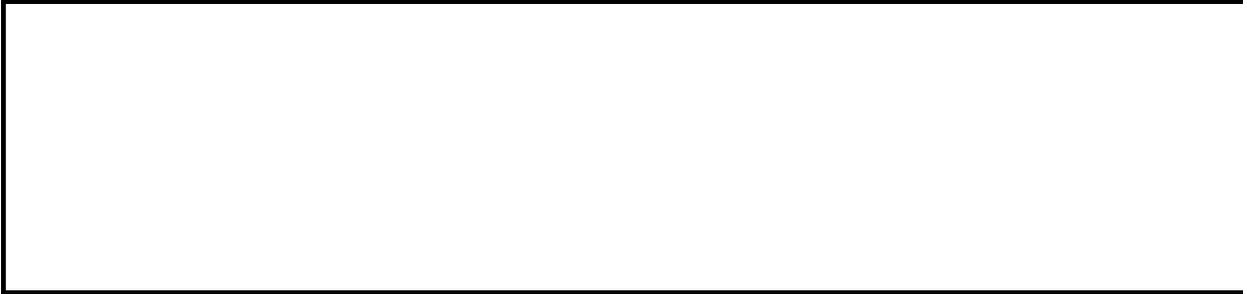
Build Property

-
Returns the version and build number of the Word application. Read-only **String**.

Example

This example displays the version and build number of Word.

```
MsgBox Prompt:=Application.Build, _  
        Title:="Microsoft Word Version"
```



BuiltIn Property

-

True if the specified object is one of the built-in styles or caption labels in Word.
Read-only **Boolean**.

Remarks

You can specify built-in styles across all languages by using the **WdBuiltinStyle** constants or within a language by using the style name for the language version of Word. For example, if you specify U.S. English in your Microsoft Office language settings, the following statements are equivalent:

```
ActiveDocument.Styles(wdStyleHeading1)
```

```
ActiveDocument.Styles("Heading 1")
```

Example

This example checks all the styles in the active document. When it finds a style that isn't built in, it displays the name of the style.

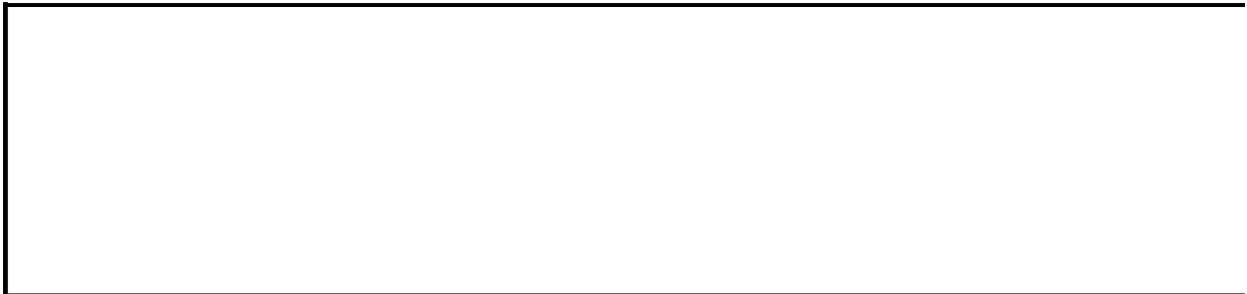
```
Dim styleLoop As Style

For Each styleLoop in ActiveDocument.Styles
    If styleLoop.BuiltIn = False Then
        MsgBox styleLoop.NameLocal
    End If
Next styleLoop
```

This example checks all the caption labels that have been created in the application. When it finds a caption label that isn't built in, it displays the name of the label.

```
Dim clLoop As CaptionLabel

For Each clLoop in CaptionLabels
    If clLoop.BuiltIn = False Then
        MsgBox clLoop.Name
    End If
Next clLoop
```



BuiltinDictionary Property

Returns a [Dictionary](#) object that represents the main dictionary Microsoft Word uses during conversion between Hangul and Hanja.

expression.**BuiltinDictionary**

expression Required. An expression that returns a [HangulHanjaConversionDictionaries](#) object.

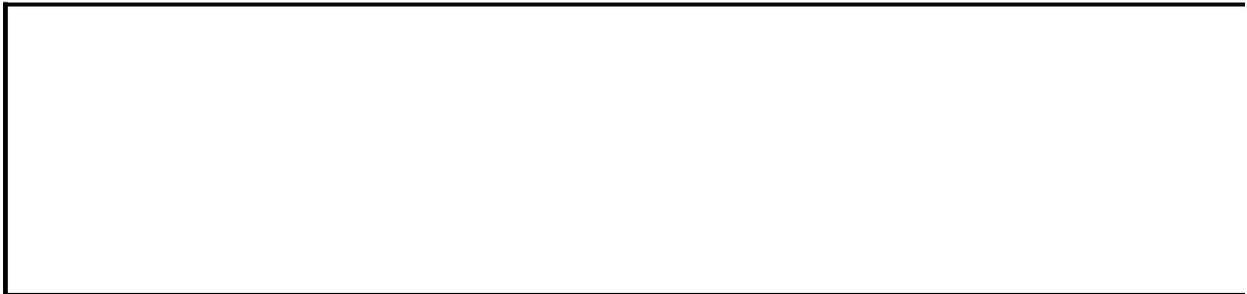
Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example displays the full path for the main Hangul-Hanja conversion dictionary.

```
With HangulHanjaDictionaries.BuiltInDictionary  
    MsgBox .Path & Application.PathSeparator & .Name  
End With
```



BuiltInDocumentProperties Property

Returns a [DocumentProperties](#) collection that represents all the built-in document properties for the specified document.

expression.**BuiltInDocumentProperties**

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

Remarks

To return a single [DocumentProperty](#) object that represents a specific built-in document property, use **BuiltinDocumentProperties(index)**, where *index* is a **WdBuiltInProperty** constant. For a list of valid constants, consult the Microsoft Visual Basic Object Browser. For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

If Microsoft Word doesn't define a value for one of the built-in document properties, reading the [Value](#) property for that document property generates an error.

Use the [CustomDocumentProperties](#) property to return the collection of custom document properties.

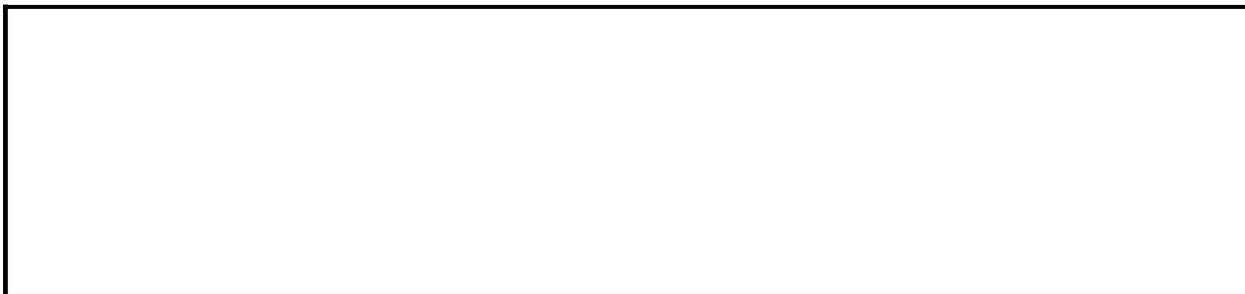
Example

This example inserts a list of built-in properties at the end of the active document.

```
Sub ListProperties()  
    Dim rngDoc As Range  
    Dim proDoc As DocumentProperty  
  
    Set rngDoc = ActiveDocument.Content  
  
    rngDoc.Collapse Direction:=wdCollapseEnd  
  
    For Each proDoc In ActiveDocument.BuiltInDocumentProperties  
        With rngDoc  
            .InsertParagraphAfter  
            .InsertAfter proDoc.Name & "= "  
            On Error Resume Next  
            .InsertAfter proDoc.Value  
        End With  
    Next  
End Sub
```

This example displays the number of words in the active document.

```
Sub DisplayTotalWords()  
    Dim intWords As Integer  
    intWords = ActiveDocument.BuiltInDocumentProperties(wdPropertyWo  
    MsgBox "This document contains " & intWords & " words."  
End Sub
```



ButtonFieldClicks Property

-

Returns or sets the number of clicks (either one or two) required to run a GOTOBUTTON or MACROBUTTON field. Read/write **Long**.

Example

This example sets the number of clicks required to run a MACROBUTTON or GOTOBUTTON field to one.

```
Options.ButtonFieldClicks = 1
```



CalculateOnExit Property

-

True if references to the specified form field are automatically updated whenever the field is exited. Read/write **Boolean**.

Remarks

A REF field can be used to reference the contents of a form field. For example, {REF SubTotal} references the form field marked by the SubTotal bookmark.

Example

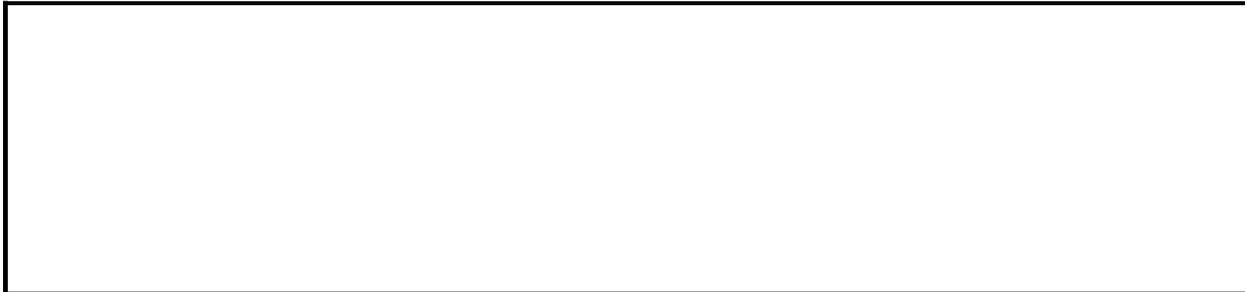
This example keeps references to form fields in Form.doc from being automatically updated whenever the form field is exited.

```
Dim ffLoop As FormField

For Each ffLoop In Documents("Form.doc").FormFields
    ffLoop.CalculateOnExit = False
Next ffLoop
```

This example adds a text form field and a REF field in a new document. Whenever text is typed and the Text1 field is exited, the REF field is automatically updated.

```
With Documents.Add
    .FormFields.Add Range:=Selection.Range, _
        Type:=wdFieldFormTextInput
    .Fields.Add Range:=Selection.Range, _
        Type:=wdFieldRef, Text:="Text1"
    .FormFields("Text1").CalculateOnExit = True
    .Protect Type:=wdAllowOnlyFormFields
End With
```



Callout Property

-

Returns a [CalloutFormat](#) object that contains callout formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent callouts. Read-only.

Example

This example adds to myDocument an oval 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.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    .AddShape msoShapeOval, 180, 200, 280, 130
    With .AddCallout(msoCalloutTwo, 420, 170, 170, 40)
        .TextFrame.TextRange.Text = "My oval"
        With .Callout
            .Accent = True
            .Border = False
        End With
    End With
End With
```



CanOpen Property

-

True if the specified file converter is designed to open files. Read-only **Boolean**.

Note The [CanSave](#) property returns **True** if the specified file converter can be used to save (export) files.

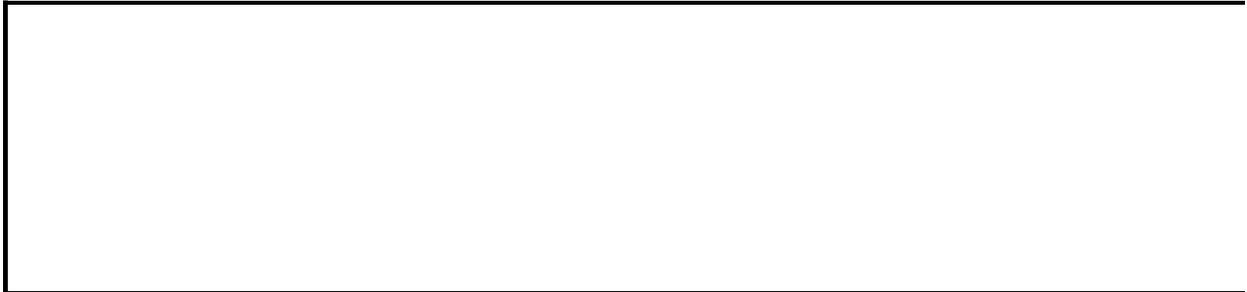
Example

This example determines whether the first file converter is able to open files.

```
If FileConverters(1).CanOpen = True Then
    MsgBox FileConverters(1).FormatName & " can open files"
End If
```

This example determines whether the WordPerfect6x file converter can be used to open files. If the **CanOpen** property returns **True**, a document named "Test.wp" is opened.

```
If FileConverters("WordPerfect6x").CanOpen = True Then
    Documents.Open FileName:="C:\Test.wp", _
        Format:=FileConverters("WordPerfect6x").OpenFormat
End If
```



CanSave Property

-
True if the specified file converter is designed to save files. Read-only **Boolean**.

Note The [CanOpen](#) property returns **True** if the specified file converter can be used to open (import) files.

Example

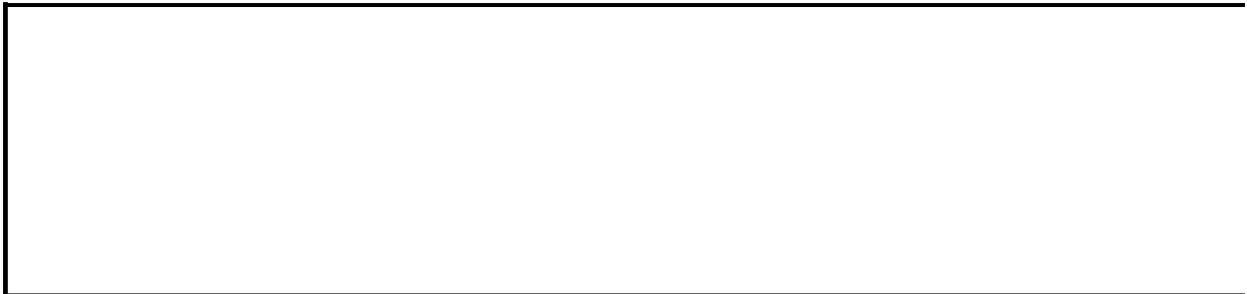
This example determines whether the WordPerfect converter can be used to save files. If the return value is **True**, the active document is saved in WordPerfect 6.x format.

```
Dim lngSaveFormat As Long

If Application.FileConverters("WordPerfect6x").CanSave = _
    True Then
    lngSaveFormat = _
        Application.FileConverters("WordPerfect6x").SaveFormat
    ActiveDocument.SaveAs FileName:="C:\Document.wp", _
        FileFormat:=lngSaveFormat
End If
```

This example displays a message that indicates whether the third converter in the **FileConverters** collection can save files.

```
If FileConverters(3).CanSave = True Then
    MsgBox FileConverters(3).FormatName & " can save files"
Else
    MsgBox FileConverters(3).FormatName & " cannot save files"
End If
```



CanvasItems Property

Returns a [CanvasShapes](#) object that represents a collection of shapes in a drawing canvas.

expression.**CanvasItems**

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

Example

This example creates a new drawing canvas in the active document and adds a circle to the canvas.

```
Sub NewCanvasShape()  
    Dim shpCanvas As Shape  
    Set shpCanvas = ActiveDocument.Shapes.AddCanvas( _  
        Left:=100, Top:=75, Width:=150, Height:=200)  
    shpCanvas.CanvasItems.AddShape _  
        Type:=msoShapeOval, Top:=25, _  
        Left:=25, Width:=150, Height:=150  
End Sub
```



CapsLock Property

-

True if the CAPS LOCK key is turned on. Read-only **Boolean**.

Example

This example retrieves the current state of the CAPS LOCK key.

```
Dim blnCapsLock As Boolean
```

```
blnCapsLock = Application.CapsLock
```



Caption Property

-

TableOfFigures object: Returns or sets the label that identifies the items to be included in a table of figures. Corresponds to the \c switch for a TOC field. Read/write **String**.

Window or **Application** object: Returns or sets the caption text for the specified document or application window. Read/write **String**.

Remarks

To change the caption of the application window to the default text, set this property to an empty string ("").

Example

This example displays the caption of each window in the **Windows** collection.

```
Count = 1
For Each win In Windows
    MsgBox Prompt:=win.Caption, Title:="Window" & Str(Count) & _
    " Caption"
    Count = Count + 1
Next win
```

This example resets the caption of the application window.

```
Application.Caption = ""
```

This example sets the caption of the active window to the active document name.

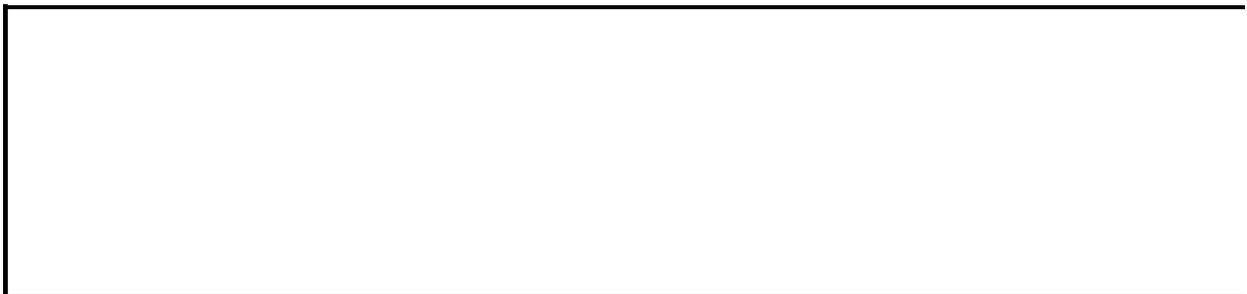
```
ActiveDocument.ActiveWindow.Caption = ActiveDocument.FullName
```

This example changes the caption of the Word application window to include the user name.

```
Application.Caption = UserName & "'s copy of Word"
```

This example inserts a Table caption and then changes the caption of the first table of figures to "Table."

```
Selection.Collapse Direction:=wdCollapseStart
Selection.Range.InsertCaption "Table"
If ActiveDocument.TablesOfFigures.Count >= 1 Then
    ActiveDocument.TablesOfFigures(1).Caption = "Table"
End If
```



CaptionLabel Property

Returns or sets the caption label ("Figure," "Table," or "Equation," for example) of the specified caption. Read/write **Variant**.

Note This property can be set to a string or a **WdCaptionLabelID** constant.

Example

This example displays the name ("Microsoft Excel Worksheet," for example) and caption label ("Figure," for example) for each item that has a caption added automatically when inserted.

```
Dim acLoop As AutoCaption

For Each acLoop In AutoCaptions
    If acLoop.AutoInsert = True Then MsgBox acLoop.Name _
        & vbCr & "Label = " & acLoop.CaptionLabel.Name
Next acLoop
```

This example sets the caption label for Word tables to "Table" and then inserts a new table immediately after the selection.

```
With AutoCaptions("Microsoft Word Table")
    .AutoInsert = True
    .CaptionLabel = wdCaptionTable
End With
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.Tables.Add Range:=Selection.Range, NumRows:=2, _
    NumColumns:=3
```



CaptionLabels Property

Returns a [CaptionLabels](#) collection that represents all the available caption labels. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the numbering style for table captions.

```
CaptionLabels(wdCaptionTable).NumberStyle = _  
    wdCaptionNumberStyleLowercaseRoman
```

This example adds a new caption label named "Photo" and then inserts a photo caption.

```
CaptionLabels.Add Name:="Photo"  
With Selection  
    .InsertParagraphAfter  
    .InsertCaption Label:="Photo"  
End With
```



↳ [Show All](#)

Case Property

Returns or sets a [WdCharacterCase](#) constant that represents the case of the text in the specified range. Read/write.

WdCharacterCase can be one of these WdCharacterCase constants.

wdFullWidth

wdHalfWidth

wdHiragana

wdKatakana

wdLowerCase

wdNextCase

wdTitleSentence

wdTitleWord

wdToggleCase

wdUpperCase

expression.Case

expression Required. An expression that returns a [Range](#) object.

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 changes the first word in the selection to uppercase.

```
Selection.Words(1).Case = wdUpperCase
```

This example capitalizes the first letter of each sentence in the first paragraph of the document.

```
Set myRange = ActiveDocument.Paragraphs(1).Range  
For Each Sent In myRange.Sentences  
    Sent.Case = wdTitleSentence  
Next Sent
```



Category Property

Returns or sets the category of entries to be included in a table of authorities. Corresponds to the \c switch for a TOA field. Values 1 through 16 correspond to the items in the **Category** list on the **Table of Authorities** tab in the **Index and Tables** dialog box. Read/write **Long**.

Note The number 0 (zero), which corresponds to all categories, cannot be used with this property. You can, however, use 0 to specify all categories when you're inserting a table of authorities. The following example inserts a table of authorities for all categories.

```
ActiveDocument.TablesOfAuthorities.Add _  
    Range:=Selection.Range, Category:=0
```

Example

This example formats the first table of authorities in the active document to include all citations in the first category (by default, the Cases category).

```
If ActiveDocument.TablesOfAuthorities.Count >= 1 Then
    ActiveDocument.TablesOfAuthorities(1).Category = 1
End If
```



CCList Property

-
Returns or sets the carbon copy (CC) recipients for a letter created by the Letter Wizard. Read/write **String**.

Example

This example displays the CC list text for the active document.

```
MsgBox ActiveDocument.GetLetterContent.CCList
```

This example creates a new **LetterContent** object, sets the courtesy copies by setting the **CCList** property, and then runs the Letter Wizard by using the **RunLetterWizard** method.

```
Dim lcNew As New LetterContent
```

```
lcNew.CCList = "K. Jordan, D. Funk, D. Morrison"  
ActiveDocument.RunLetterWizard LetterContent:=lcNew
```



Cells Property

-
Returns a [Cells](#) collection that represents the table cells in a column, row, selection, or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a 3x3 table and assigns a sequential cell number to each cell in the table.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Selection.Range, 3, 3)
i = 1
For Each c In myTable.Range.Cells
    c.Range.InsertAfter "Cell " & i
    i = i + 1
Next c
```

This example sets the current cell's background color to red.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells(1).Shading.BackgroundPatternColorIndex = wdRed
Else
    MsgBox "The insertion point is not in a table."
End If
```



ChapterPageSeparator Property

Returns or sets the separator character used between the chapter number and the page number. Can be one of the following read/write [WdSeparatorType](#) constants.

WdSeparatorType can be one of these WdSeparatorType constants.

wdSeparatorColon

wdSeparatorEnDash

wdSeparatorPeriod

wdSeparatorEmDash

wdSeparatorHyphen

expression.**ChapterPageSeparator**

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

Remarks

Before you can create page numbers that include chapter numbers, the document headings must have a numbered outline format applied that uses styles from the **Bullets and Numbering** dialog box. To do this in Visual Basic, use the **ApplyListTemplate** method.

Example

The first part of this example creates a new document, adds chapter titles and page breaks, and then formats the document by using the last numbered outline format listed in the **Bullets and Numbering** dialog box. The second part of the example adds centered page numbers — including the chapter number — to the header; an en dash separates the chapter number and the page number.

```
Dim intLoop As Integer
Dim hfTemp As HeaderFooter

Documents.Add
For intLoop = 1 To 5
    With Selection
        .TypeParagraph
        .InsertBreak
    End With
Next intLoop
ActiveDocument.Content.Style = wdStyleHeading1
ActiveDocument.Content.ListFormat.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(7)

Set hfTemp = ActiveDocument.Sections(1) _
    .Headers(wdHeaderFooterPrimary)
With hfTemp.PageNumbers
    .Add PageNumberAlignment:=wdAlignPageNumberCenter
    .NumberStyle = wdPageNumberStyleArabic
    .IncludeChapterNumber = True
    .HeadingLevelForChapter = 0
    .ChapterPageSeparator = wdSeparatorEnDash
End With
```



ChapterStyleLevel Property

Returns or sets the heading style that marks a new chapter when chapter numbers are included with the specified caption label. The number 1 corresponds to Heading 1, 2 corresponds to Heading 2, and so on. Read/write **Long**.

Note The [IncludeChapterNumber](#) property must be set to **True** for chapter numbers to be included with caption labels.

Example

This example formats the table's caption label to include a chapter number. The chapter number is taken from paragraphs formatted with the Heading 2 style.

```
With CaptionLabels(wdCaptionTable)
    .IncludeChapterNumber = True
    .ChapterStyleLevel = 2
End With
```



Characters Property

-
Returns a [Characters](#) collection that represents the characters in a document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the first character in the selection. If nothing is selected, the character immediately after the insertion point is displayed.

```
char = Selection.Characters(1).Text  
MsgBox "The first character is... " & char
```

This example returns the number of characters in the first sentence in the active document (spaces are included in the count).

```
numchars = ActiveDocument.Sentences(1).Characters.Count
```



CharacterUnitFirstLineIndent Property

Returns or sets the value (in characters) for a first-line or hanging indent. Use a positive value to set a first-line indent, and use a negative value to set a hanging indent. Read/write **Single**.

expression.**CharacterUnitFirstLineIndent**

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

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets a first-line indent of one character for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1) _  
    .CharacterUnitFirstLineIndent = 1
```

This example sets a hanging indent of 1.5 characters for the second paragraph in the active document.

```
ActiveDocument.Paragraphs(2) _  
    .CharacterUnitFirstLineIndent = -1.5
```



CharacterUnitLeftIndent Property

Returns or sets the left indent value (in characters) for the specified paragraphs.
Read/write **Single**.

expression.**CharacterUnitLeftIndent**

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

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets the left indent of the first paragraph in the active document to one character from the left margin.

```
ActiveDocument.Paragraphs(1) _  
    .CharacterUnitLeftIndent = 1
```



CharacterUnitRightIndent Property

Returns or sets the right indent value (in characters) for the specified paragraphs.
Read/write **Single**.

expression.**CharacterUnitRightIndent**

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

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets the right indent for all paragraphs in the active document to one character from the right margin.

```
ActiveDocument.Paragraphs _  
    .CharacterUnitRightIndent = 1
```



↳ [Show All](#)

CharacterWidth Property

Returns or sets the character width of the specified range. Read/write [WdCharacterWidth](#).

WdCharacterWidth can be one of these WdCharacterWidth constants.

wdWidthFullWidth

wdWidthHalfWidth

expression.**CharacterWidth**

expression Required. An expression that returns a [Range](#) object.

Example

This example converts the current selection to half-width characters.

```
Selection.Range.CharacterWidth = wdWidthHalfWidth
```



CharsLine Property

-

Returns or sets the number of characters per line in the document grid.
Read/write **Single**.

Example

This example sets the number of characters per line to 42 for the active document.

```
ActiveDocument.PageSetup.CharsLine = 42
```



CheckBox Property

-
Returns a [CheckBox](#) object that represents a check box form field. Read-only.

Remarks

If the **CheckBox** property is applied to a **FormField** object that isn't a check box form field, the property won't fail, but the [Valid](#) property for the returned object will be **False**.

Example

This example clears the check box named "Blue."

```
ActiveDocument.FormFields("Blue").CheckBox.Value = False
```

This example compares the current value with the default value of the check box named "Check1." If the values are equal, the `blnSame` variable is set to **True**.

```
Dim ffTemp As FormField
Dim blnSame As Boolean

Set ffTemp = ActiveDocument.FormFields("Check1").CheckBox
If ffTemp.Default = ffTemp.Value Then
    blnSame = True
Else
    blnSame = False
End If
```



CheckGrammarAsYouType Property

-
True if Word checks grammar and marks errors automatically as you type.
Read/write **Boolean**.

Remarks

This property marks grammatical errors, but to see them on screen, you must set the **ShowGrammaticalErrors** property to **True**.

Example

This example sets Word to check for grammatical errors as you type and to display any errors found in the active document.

```
Options.CheckGrammarAsYouType = True  
ActiveDocument.ShowGrammaticalErrors = True
```

This example returns the status of the **Check grammar as you type** option on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

```
Dim blnCheck As Boolean
```

```
blnCheck = Options.CheckGrammarAsYouType
```



CheckGrammarWithSpelling Property

-
True if Word checks grammar while checking spelling. Read/write **Boolean**.

Remarks

This property controls whether Word checks grammar when you check spelling by using the **Spelling** command (**Tools** menu).

To check spelling or grammar from a Visual Basic procedure, use the **CheckSpelling** method to check only spelling and use the **CheckGrammar** method to check both grammar and spelling.

Example

This example returns the status of the **Check grammar with spelling** option on the **Spelling & Grammar** tab in the **Options** dialog box. If the option is selected, the procedure checks both spelling and grammar for the active document; otherwise, only spelling is checked.

```
If Options.CheckGrammarWithSpelling = True Then
    ActiveDocument.CheckGrammar
Else
    ActiveDocument.CheckSpelling
End If
```



CheckHangulEndings Property

-
True if Microsoft Word automatically detects Hangul endings and ignores them during conversion from Hangul to Hanja. Read/write **Boolean**.

expression.**CheckHangulEndings**

expression Required. An expression that returns an [Options](#) object.

Remarks

If converting from Hanja to Hangul, this property is ignored.

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example asks the user whether to set Microsoft Word to automatically detect Hangul endings and ignore them during conversion from Hangul to hanja.

```
x = MsgBox("Check Hangul endings during " _  
    & "conversion from Hangul to Hanja?", vbYesNo)  
If x = vbYes Then  
    Options.CheckHangulEndings = True  
Else  
    Options.CheckHangulEndings = False  
End If
```



CheckIfOfficeIsHTMLEditor Property

True if Microsoft Word checks to see whether an Office application is the default HTML editor when you start Word. **False** if Word does not perform this check. The default value is **True**. Read/write **Boolean**.

Remarks

This property is used only if the Web browser you are using supports HTML editing and HTML editors.

To use a different HTML editor, you must set this property to **False** and then register the editor as the default system HTML editor.

Example

This example causes Microsoft Word not to check to see whether an Office application is the default HTML editor.

```
Application.DefaultWebOptions _  
    .CheckIfOfficeIsHTMLEditor = False
```



CheckIfWordIsDefaultHTMLEditor Property

True if Microsoft Word checks to see whether it is the default HTML editor when you start Word. **False** if Word does not perform this check. The default value is **True**. Read/write **Boolean**.

Remarks

This property is used only if the Web browser you are using supports HTML editing and HTML editors.

To use a different HTML editor, you must set this property to **False** and then register the editor as the default system HTML editor.

Example

This example sets Microsoft Word to check to see whether it is the default HTML editor.

```
Application.DefaultWebOptions _  
    .CheckIfWordIsDefaultHTMLEditor = True
```



CheckLanguage Property

-
True if Microsoft Word automatically detects the language you are using as you type. Read/write **Boolean**.

expression.**CheckLanguage**

expression Required. An expression that returns an [Application](#) object.

Remarks

If you haven't set up Word for multilingual editing, the **CheckLanguage** property always returns **False**. For more information about automatic language detection, see [About automatic language detection](#).

Example

This example checks to see if automatic language detection has been activated.

```
If Application.CheckLanguage = True Then  
    MsgBox "Automatic language detection is activated!"  
End If
```



CheckSpellingAsYouType Property

-
True if Word checks spelling and marks errors automatically as you type.
Read/write **Boolean**.

Remarks

This property marks spelling errors, but to see them on the screen, you must set the **ShowSpellingErrors** property to **True**.

Example

This example turns off automatic spell checking in Word.

```
Options.CheckSpellingAsYouType = False
```

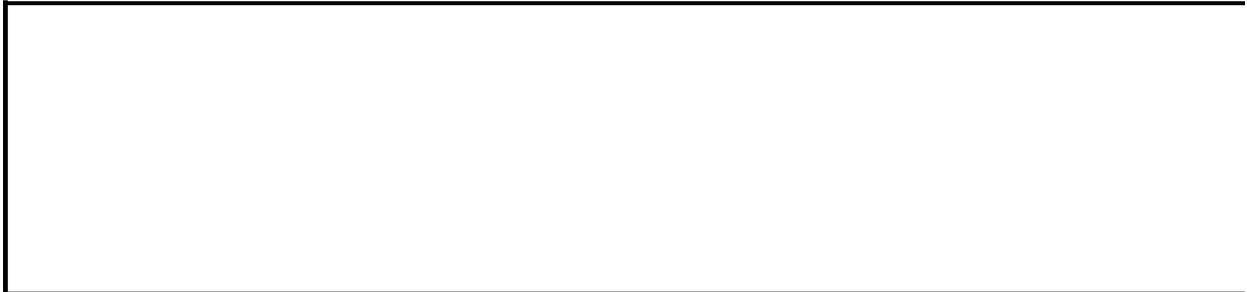
This example sets Word to check for spelling errors as you type and to display any errors found in the active document.

```
Options.CheckSpellingAsYouType = True  
ActiveDocument.ShowSpellingErrors = True
```

This example returns the status of the **Check spelling as you type** option on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

```
Dim blnCheck As Boolean
```

```
blnCheck = Options.CheckSpellingAsYouType
```



↳ [Show All](#)

Child Property

-
True if the shape is a child shape or if all shapes in a shape range are child shapes of the same parent. Read-only [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

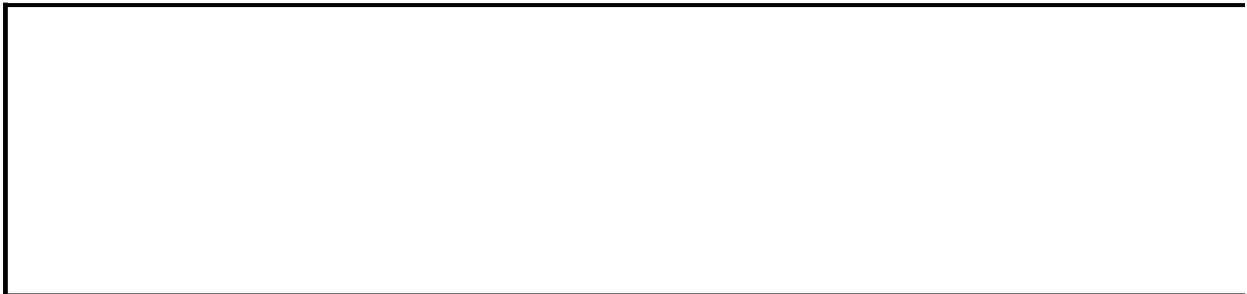
expression.**Child**

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

Example

This example selects the first shape in the canvas and, if the selected shape is a child shape, fills the shape with the specified color. This example assumes that the first shape in the active document is a drawing canvas that contains multiple shapes.

```
Sub FillChildShape()  
  
    Dim shpCanvasItem As Shape  
  
    'Select the first shape in the drawing canvas  
    Set shpCanvasItem = ActiveDocument.Shapes(1).CanvasItems(1)  
  
    'Fill selected shape if it is a child shape  
    With shpCanvasItem  
        If .Child = msoTrue Then  
            .Fill.ForeColor.RGB = RGB(100, 0, 200)  
        Else  
            MsgBox "This shape is not a child shape."  
        End If  
    End With  
  
End Sub
```



ChildFramesetCount Property

-

Returns the number of child **Frameset** objects associated with the specified **Frameset** object. This property applies only to **Frameset** objects of type **wdFramesetTypeFrameset**. Read-only **Long**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example displays the number of child **Frameset** objects contained by the **Frameset** object that represents the specified frames page.

```
MsgBox ActiveWindow.Document_  
    .Frameset.ChildFramesetCount
```



ChildFramesetItem Property

Returns the [Frameset](#) object that represents the child **Frameset** object specified by the **Index** argument. This property applies only to **Frameset** objects of type **wdFramesetTypeFrameset**. Read-only.

expression.**ChildFramesetItem**(**Index**)

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

Index Required **Long**. The index number of the specified frame.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the name of the third child frame of the specified frame to "BottomFrame".

```
ActiveWindow.Document.Frameset _  
    .ChildFramesetItem(3).FrameName = "BottomFrame"
```



Children Property

Returns a **DiagramNodeChildren** object that contains all of the children of the calling diagram node.

expression.Children

expression Required. An expression that returns a [DiagramNode](#) object.

Example

The following example creates a diagram and adds child nodes to it.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram( _  
        Type:=msoDiagramPyramid, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add first child diagram node  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add four more child nodes  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
End Sub
```



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 document with a drawing canvas, populates the drawing canvas with shapes, and then, after checking that the shapes selected are child shapes, fills the child shapes with a pattern.

```
Sub ChildShapes()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    'Create a new document with a drawing canvas and shapes  
    Set docNew = Documents.Add  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=100, Top:=100, Width:=200, Height:=200)  
    shpCanvas.CanvasItems.AddShape msoShapeRectangle, _  
        Left:=0, Top:=0, Width:=100, Height:=100  
    shpCanvas.CanvasItems.AddShape msoShapeOval, _  
        Left:=0, Top:=50, Width:=100, Height:=100  
    shpCanvas.CanvasItems.AddShape msoShapeDiamond, _  
        Left:=0, Top:=100, Width:=100, Height:=100  
  
    'Select all shapes in the canvas  
    shpCanvas.CanvasItems.SelectAll  
  
    'Fill canvas child shapes with a pattern  
    If Selection.HasChildShapeRange = True Then  
        Selection.ChildShapeRange.Fill.Patterned msoPatternDivot  
    Else  
        MsgBox "This is not a range of child shapes."  
    End If  
  
End Sub
```



ClassName Property

-

Returns a unique name that identifies the file converter. Read-only **String**.

Example

This example displays the class name and format name of the first converter in the **FileConverters** collection.

```
MsgBox "ClassName= " & FileConverters(1).ClassName & vbCr _  
      & "FormatName= " & FileConverters(1).FormatName
```

If an HTML file converter is available, this example sets the HTML format as the default save format.

```
Dim fcLoop As FileConverter  
  
For Each fcLoop In FileConverters  
    If fcLoop.ClassName = "HTML" Then _  
        Application.DefaultSaveFormat = "HTML"  
Next fcLoop
```



ClassType Property

-

Returns or sets the class type for the specified OLE object, picture, or field.
Read/write **String**.

Remarks

This property is read-only for linked objects other than DDE links.

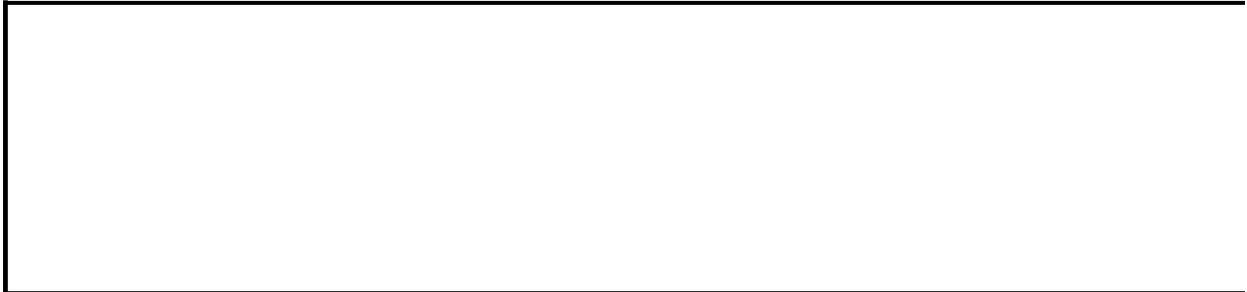
You can see a list of the available applications in the **Object type** box on the **Create New** tab in the **Object** dialog box (**Insert** menu). You can find the **ClassType** string by inserting an object as an inline shape and then viewing the field codes. The class type of the object follows either the word "EMBED" or the word "LINK."

Example

This example loops through all the floating shapes on the active document and sets all linked Microsoft Excel worksheets to be updated automatically.

```
Dim shapeLoop As Shape

For Each shapeLoop In ActiveDocument.Shapes
    With shapeLoop
        If .Type = msoLinkedOLEObject Then
            If .OLEFormat.ClassType = "Excel.Sheet" Then
                .LinkFormat.AutoUpdate = True
            End If
        End If
    End With
Next
```



ClickAndTypeParagraphStyle Property

Returns or sets the default paragraph style applied to text by the Click and Type feature in the specified document. To set this property, specify either the local name of the style, an integer, or a **WdBuiltInStyle** constant, or an object that represents the style. Read/write **Variant**.

expression.**ClickAndTypeParagraphStyle**

expression Required. An expression that returns a [Document](#) object.

Remarks

For a list of the **WdBuiltinStyle** constants, consult the Microsoft Visual Basic Object Browser.

If the [InUse](#) property for the specified style is set to **False**, an error occurs.

For more information on Click and Type, see [Overview of Click and Type](#).

Example

This example sets the default paragraph style applied by Click and Type to Plain Text.

```
With ActiveDocument
  x = "Plain Text"
  If .Styles(x).InUse Then
    .ClickAndTypeParagraphStyle = x
  Else
    MsgBox "Sorry, this style is not in use yet."
  End If
End With
```



Closing Property

-

Returns or sets the closing text for a letter created by the Letter Wizard (for example, "Sincerely yours"). Read/write **String**.

Example

This example displays the closing text from the active document.

```
MsgBox ActiveDocument.GetLetterContent.Closing
```

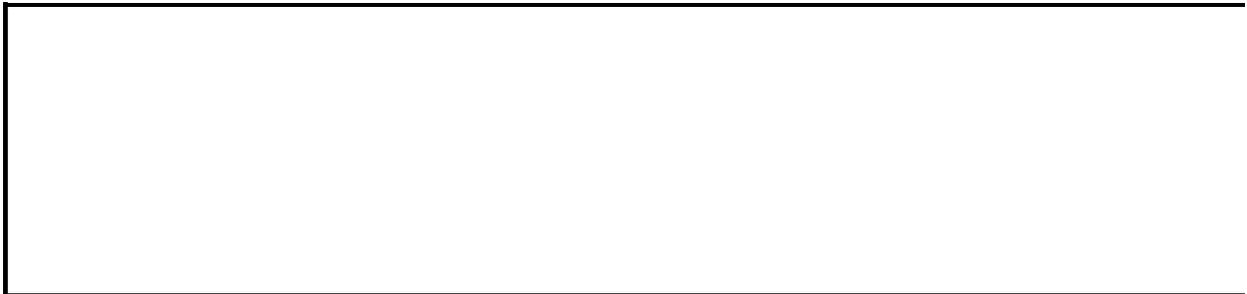
This example retrieves letter elements from the active document, changes the closing text by setting the **Closing** property, and then uses the **SetLetterContent** method to update the document to reflect the changes.

```
Dim lcCurrent As LetterContent
```

```
Set lcCurrent = ActiveDocument.GetLetterContent
```

```
lcCurrent.Closing = "Sincerely yours,"
```

```
ActiveDocument.SetLetterContent LetterContent:=lcCurrent
```



Code Property

-

Returns a [Range](#) object that represents a field's code. A field's code is everything that's enclosed by the field characters ({ }) including the leading space and trailing space characters. You can access a field's code without changing the view from field results. Read/write.

Example

This example displays the field code for each field in the active document.

```
Dim fieldLoop As Field

For Each fieldLoop In ActiveDocument.Fields
    MsgBox Chr(34) & fieldLoop.Code.Text & Chr(34)
Next fieldLoop
```

This example changes the field code for the first field in the active document to CREATEDATE.

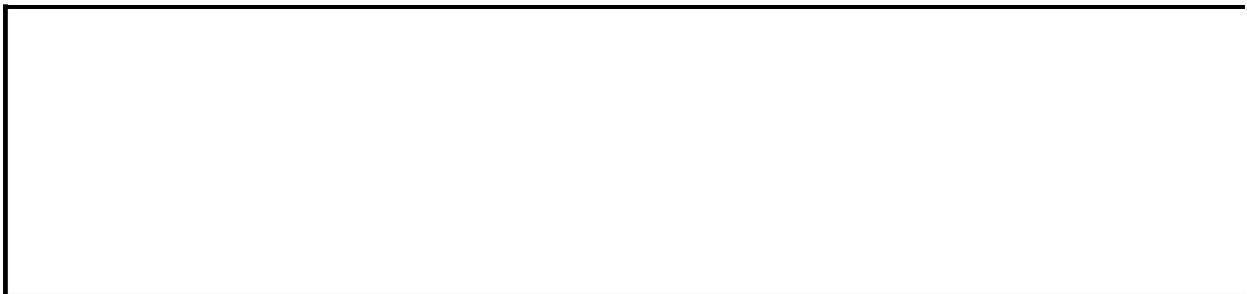
```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Fields(1).Code
rngTemp.Text = " CREATEDATE "
ActiveDocument.Fields(1).Update
```

This example determines whether the active document includes a mail merge field named "Title."

```
Dim fieldLoop As Field

For Each fieldLoop In ActiveDocument.MailMerge.Fields
    If InStr(1, fieldLoop.Code.Text, "Title", 1) Then
        MsgBox "A Title merge field is in this document"
    End If
Next fieldLoop
```



CodeName Property

-

Returns the code name for the specified document. Read-only **String**.

Remarks

The code name is the name for the module that houses event macros for a document. The default name for the module is "ThisDocument"; you can view it in the **Project** window. For information about using events with the **Document** object, see [Using Events with the Document Object](#).

Example

This example returns the name of the code window for the active document.

```
Msgbox ActiveDocument.CodeName
```



↳ [Show All](#)

Color Property

Returns or sets the 24-bit color for the specified [Border](#) or [Font](#) object. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**Color**

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

Example

This example changes the color of the text in the first paragraph of the active document to green.

```
ActiveDocument.Paragraphs(1).Range.Font.Color = wdColorGreen
```

This example changes the color of the selected text to dark red.

```
Selection.Font.Color = wdColorDarkRed
```

This example adds a dotted indigo border around each cell in the first table.

```
If ActiveDocument.Tables.Count >= 1 Then
    For Each aBorder In ActiveDocument.Tables(1).Borders
        aBorder.Color = wdColorIndigo
        aBorder.LineStyle = wdLineStyleDashDot
        aBorder.LineWidth = wdLineWidth075pt
    Next aBorder
End If
```



ColorIndex Property

Returns or sets the color for the specified border or font object. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**ColorIndex**

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

Remarks

The **wdByAuthor** constant is not valid for border and font objects.

Example

This example changes the color of the text in the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Range.Font.ColorIndex = wdGreen
```

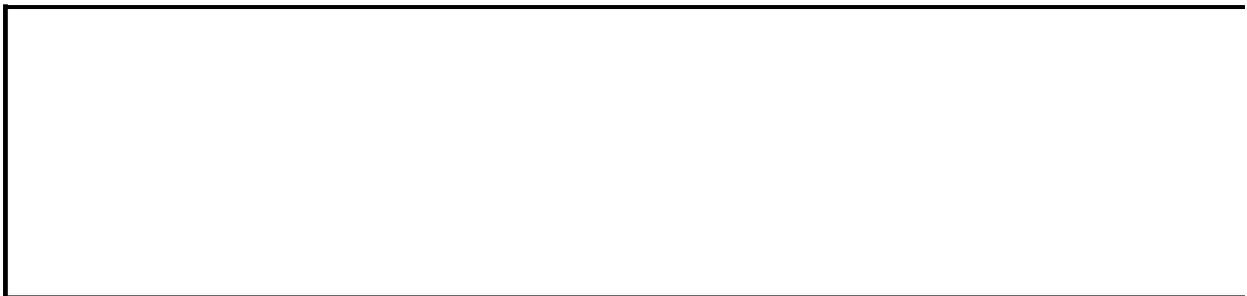
This example formats the selected text to appear in red.

```
Selection.Font.ColorIndex = wdRed
```

This example adds a dotted red border around each cell in the first table.

```
Dim borderLoop As Border
```

```
If ActiveDocument.Tables.Count >= 1 Then
    For Each borderLoop In ActiveDocument.Tables(1).Borders
        With borderLoop
            .ColorIndex = wdRed
            .LineStyle = wdLineStyleDashDot
            .LineWidth = wdLineWidth075pt
        End With
    Next borderLoop
End If
```



↳ [Show All](#)

ColorIndexBi Property

Returns or sets the color for the specified [Font](#) object in a right-to-left language document. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**ColorIndexBi**

expression Required. An expression that returns a [Font](#) object.

Remarks

The **wdByAuthor** constant is not valid for **Font** objects.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the color of the **Font** object to teal.

```
Selection.Font.ColorIndexBi = wdTeal
```



ColorType Property

Returns or sets the type of color transformation applied to the specified picture or OLE object. Read/write [MsoPictureColorType](#).

MsoPictureColorType can be one of these MsoPictureColorType constants.

msoPictureAutomatic

msoPictureBlackAndWhite

msoPictureGrayscale

msoPictureMixed

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 on the active document. The first shape must be either a picture or an OLE object.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
docActive.Shapes(1).PictureFormat.ColorType = _  
    msoPictureGrayscale
```



↳ [Show All](#)

Column Property

▶ [_Column property as it applies to the **Bookmark** object.](#)

True if the specified bookmark is a table column. Read-only **Boolean**.

expression.**Column**

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

▶ [_Column property as it applies to the **Cell** object.](#)

Returns a read-only **Column** object that represents the table column containing the specified cell.

expression.**Column**

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

Example

▶ [As it applies to the **Bookmark** object.](#)

This example creates a table with a bookmark and then displays a message box that confirms that the bookmark is a table column.

```
Dim docNew As Document
Dim tableNew As Table
Dim rangeCell As Range

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 3, 5)
Set rangeCell = tableNew.Cell(3,5).Range

rangeCell.InsertAfter "Cell(3,5)"
docNew.Bookmarks.Add Name:="BKMK_Cell35", Range:=rangeCell
MsgBox docNew.Bookmarks(1).Column
```

▶ [As it applies to the **Cell** object.](#)

This example creates a 3x5 table and applies shading to the even-numbered columns.

```
Dim tableNew As Table
Dim cellLoop As Cell

Selection.Collapse Direction:=wdCollapseStart
Set tableNew = _
    ActiveDocument.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=5)

For Each cellLoop In tableNew.Rows(1).Cells
    If cellLoop.ColumnIndex Mod 2 = 0 Then
        cellLoop.Column.Shading.Texture = wdTexture10Percent
    End If
Next cellLoop
```



ColumnIndex Property

-

Returns the number of the table column that contains the specified cell. Read-only **Long**.

Example

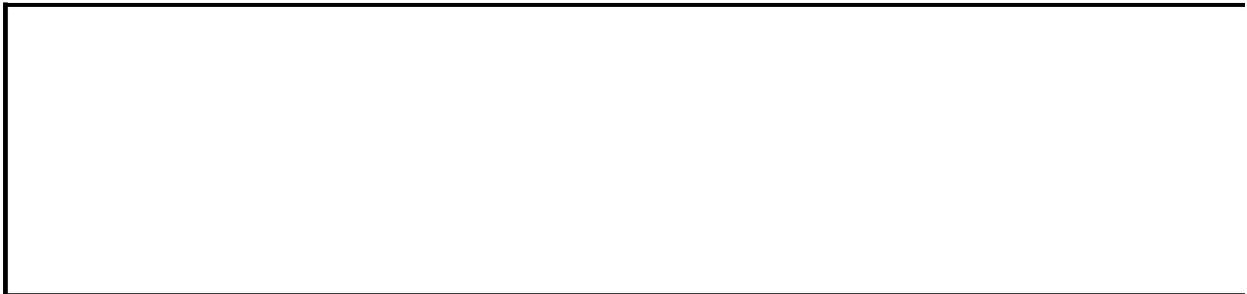
This example creates a table in a new document, selects each cell in the first row, and returns the column number that contains the selected cell.

```
Dim docNew As Document
Dim tableNew As Table
Dim cellLoop As Cell

Set docNew = Documents.Add
Set tableNew = docNew.Tables.Add(Selection.Range, 3, 3)
For Each cellLoop In tableNew.Rows(1).Cells
    cellLoop.Select
    MsgBox "This is column " & cellLoop.ColumnIndex
Next cellLoop
```

This example returns the column number of the cell that contains the insertion point.

```
Msgbox Selection.Cells(1).ColumnIndex
```



Columns Property

Returns a [Columns](#) collection that represents all the table columns in the range, selection, or table. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the number of columns in the first table in the active document.

```
If ActiveDocument.Tables.Count >= 1 Then
    MsgBox ActiveDocument.Tables(1).Columns.Count
End If
```

This example sets the width of the current column to 1 inch.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Columns.SetWidth ColumnWidth:=InchesToPoints(1), _
        RulerStyle:=wdAdjustProportional
End If
```



ColumnSelectMode Property

-
True if column selection mode is active. When this mode is active, the letters "COL" appear on the status bar. Read/write **Boolean**.

Example

This example selects a column of text that's two words across and three lines deep. The example copies the selection to the Clipboard and cancels column selection mode.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .ColumnSelectMode = True
    .MoveRight Unit:=wdWord, Count:=2, Extend:=wdExtend
    .MoveDown Unit:=wdLine, Count:=2, Extend:=wdExtend
    .Copy
    .ColumnSelectMode = False
End With
```



ColumnStripe Property

Returns or sets a **Long** that represents the number of columns in the banding when a style specifies odd- or even-column banding. Read/write.

expression.**ColumnStripe**

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

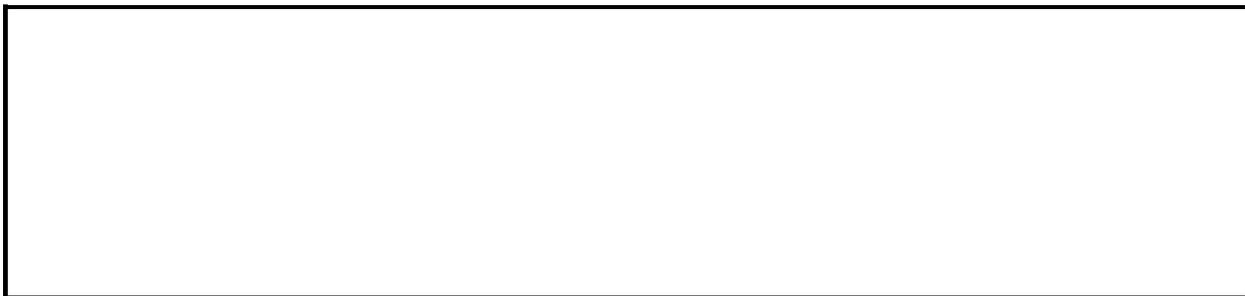
Remarks

Use the [Condition](#) method to set odd- or even-column banding for a table style.

Example

This example creates and formats a new table style then applies the new style to a new table. The resulting style causes three columns every third column and two rows every second row to have 20% shading.

```
Sub NewTableStyle()  
    Dim styTable As Style  
  
    With ActiveDocument  
        Set styTable = .Styles.Add(Name:="TableStyle 1", _  
            Type:=wdStyleTypeTable)  
  
        With .Styles("TableStyle 1").Table  
            .Condition(wdEvenColumnBanding).Shading _  
                .Texture = wdTexture20Percent  
            .ColumnStripe = 3  
            .Condition(wdEvenRowBanding).Shading _  
                .Texture = wdTexture20Percent  
            .RowStripe = 2  
        End With  
  
        With .Tables.Add(Range:=Selection.Range, NumRows:=15, _  
            NumColumns:=15)  
            .Style = ActiveDocument.Styles("TableStyle 1")  
        End With  
    End With  
  
End Sub
```



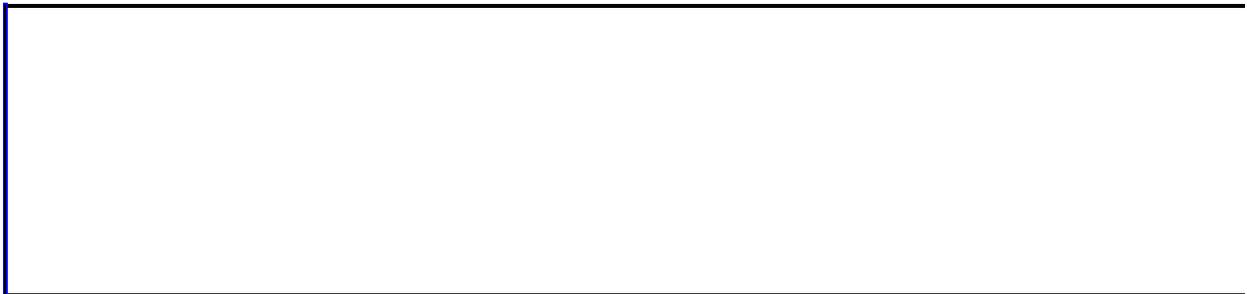
↳ [Show All](#)

COMAddIns Property

Returns a reference to the [COMAddIns](#) collection that represents all the [Component Object Model \(COM\) add-ins](#) currently loaded in Microsoft Word. These are listed in the **COM Add-Ins** dialog box. You can add the **COM Add-Ins** command to your **Tools** menu by using the **Customize** dialog box (**Tools** menu).

expression.COMAddIns

expression Required. An expression that returns an [Application](#) object.



CombineCharacters Property

True if the specified range contains combined characters. Read/write **Boolean**.

expression.CombineCharacters

expression Required. An expression that returns a [Range](#) object.

Example

This example combines the characters in the selected range.

```
Selection.Range.CombineCharacters = True
```



Command Property

-

Returns the command assigned to the specified key combination. Read-only **String**.

Example

This example displays the keys assigned to font names. A message is displayed if no keys have been assigned to fonts.

```
Dim kbLoop As KeyBinding
```

```
For Each kbLoop In KeyBindings
```

```
    If kbLoop.KeyCategory = wdKeyCategoryFont Then
```

```
        Count = Count + 1
```

```
        MsgBox kbLoop.Command & vbCr & kbLoop.KeyString
```

```
    End If
```

```
Next kbLoop
```

```
If Count = 0 Then MsgBox "Keys haven't been assigned to fonts"
```



CommandBars Property

Returns a [CommandBars](#) collection that represents the menu bar and all the toolbars in Microsoft Word.

expression.**CommandBars**

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

Remarks

Use the [CustomizationContext](#) property to set the template or document context prior to accessing the **CommandBars** collection.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example enlarges all command bar buttons and enables ToolTips.

```
With CommandBars  
    .LargeButtons = True  
    .DisplayTooltips = True  
End With
```

This example displays the **Drawing** toolbar at the bottom of the application window.

```
With CommandBars("Drawing")  
    .Visible = True  
    .Position = msoBarBottom  
End With
```

This example adds the **Versions** command button to the **Standard** toolbar.

```
CustomizationContext = NormalTemplate  
CommandBars("Standard").Controls.Add Type:=msoControlButton, _  
    ID:=2522, Before:=4
```



CommandName Property

-

Returns the name of the procedure that displays the specified built-in dialog box.
Read-only **String**.

Remarks

For more information about working with built-in Word dialog boxes, see [Displaying built-in Word dialog boxes](#).

Example

This example displays the name of the procedure that displays the **Save As** dialog box (**File** menu), FileSaveAs.

```
MsgBox Dialogs(wdDialogFileSaveAs).CommandName
```



CommandParameter Property

Returns the command parameter assigned to the specified shortcut key. Read-only **String**.

Note For information about commands that take parameters, see [Add Method \(KeyBindings Object\)](#). Use the **Command** property to return the command name assigned to the specified shortcut key.

Example

This example assigns a shortcut key to the **FontSize** command, with a command parameter of 8. Use the **CommandParameter** property to display the command parameter along with the command name and key string.

```
Dim kbNew As KeyBinding

Set kbNew = KeyBindings.Add(KeyCategory:=wdKeyCategoryCommand, _
    Command:="FontSize", _
    KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyS), _
    CommandParameter:="8")
MsgBox kbNew.Command & Chr$(32) & kbNew.CommandParameter _
    & vbCr & kbNew.KeyString
```



Comment Property

-

Returns the comment associated with the specified version of a document. Read-only **String**.

Example

This example displays the comment text for the first version of the active document.

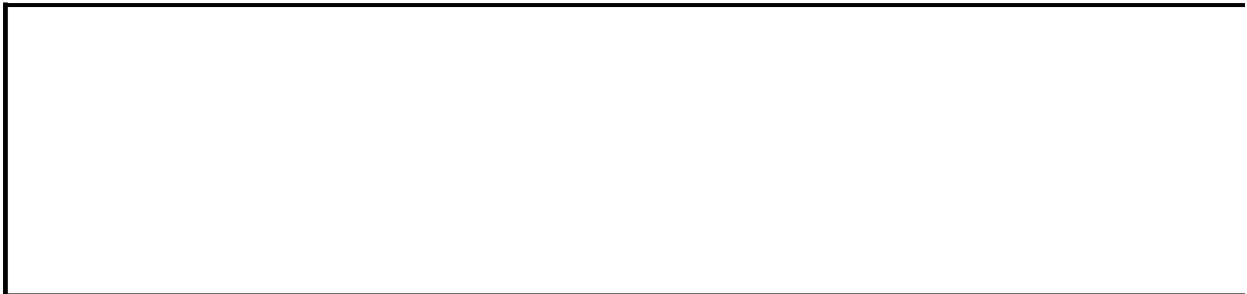
```
If ActiveDocument.Versions.Count >= 1 Then
    MsgBox Prompt:=ActiveDocument.Versions(1).Comment, _
        Title:="First Version Comment"
End If
```

This example saves a version of the document with the user's comment and then displays the comment.

```
Dim verTemp As Versions
Dim strComment As String
Dim lngCount As Long

Set verTemp = ActiveDocument.Versions

strComment = InputBox("Type a comment")
verTemp.Save Comment:=strComment
lngCount = verTemp.Count
MsgBox Prompt:=verTemp(lngCount).Comment, _
    Title:=verTemp(lngCount).SavedBy
```



Comments Property

-

Returns a [Comments](#) collection that represents all the comments in the specified document, selection, or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a comment to the selected text.

```
ActiveDocument.ActiveWindow.View.ShowHiddenText = True  
Selection.Comments.Add Range:=Selection.Range, Text:="Approved"
```

This example compares the author name of each comment in the active document with the user name on the **User Information** tab in the **Options** dialog box (**Tools** menu). If the names aren't the same, the comment reference mark is formatted to appear in red.

```
For Each comm In ActiveDocument.Comments  
    If comm.Author <> Application.UserName Then _  
        comm.Reference.Font.ColorIndex = wdRed  
Next comm
```



↳ [Show All](#)

CommentsColor Property

Returns or sets a [WdColorIndex](#) constant that represents the color of comments in a document. Read/write.

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**CommentsColor**

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

Example

This example sets the global option for Microsoft Word to color comments made in documents according to the author of the comment.

```
Sub ColorCodeComments()  
    Options.CommentsColor = wdByAuthor  
End Sub
```



↳ [Show All](#)

Compatibility Property

True if the compatibility option specified by the *Type* argument is enabled. Compatibility options affect how a document is displayed in Microsoft Word. Read/write **Boolean**.

expression.Compatibility(*Type*)

expression Required. An expression that returns a [Document](#) object.

Type Required [WdCompatibility](#). The compatibility option.

WdCompatibility can be one of these WdCompatibility constants.

wdAlignTablesRowByRow

wdApplyBreakingRules

wdAutospaceLikeWW7

wdConvMailMergeEsc

wdDontAdjustLineHeightInTable

wdDontBalanceSingleByteDoubleByteWidth

wdDontBreakWrappedTables

wdDontSnapTextToGridInTableWithObjects

wdDontULTrailSpace

wdDontUseHTMLParagraphAutoSpacing

wdExactOnTop

wdExpandShiftReturn

wdFootnoteLayoutLikeWW8

wdForgetLastTabAlignment

wdLayoutRawTableWidth

wdLayoutTableRowsApart

wdLeaveBackslashAlone

wdLineWrapLikeWord6

wdMWSmallCaps
wdNoColumnBalance
wdNoExtraLineSpacing
wdNoLeading
wdNoSpaceForUL
wdNoSpaceRaiseLower
wdNoTabHangIndent
wdOrigWordTableRules
wdPrintBodyTextBeforeHeader
wdPrintColBlack
wdSelectFieldWithFirstOrLastCharacter
wdShapeLayoutLikeWW8
wdShowBreaksInFrames
wdSpacingInWholePoints
wdSubFontBySize
wdSuppressBottomSpacing
wdSuppressSpBfAfterPgBrk
wdSuppressTopSpacing
wdSuppressTopSpacingMac5
wdSwapBordersFacingPages
wdTransparentMetafiles
wdUsePrinterMetrics
wdWPJustification
wdWrapTrailSpaces
wdTruncateFontHeight
wdUseWord97LineBreakingRules
wdWPSpaceWidth
wdWW6BorderRules

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 enables the **Suppress Space Before after a hard page or column break** option on the **Compatibility** tab in the **Options** dialog box (**Tools** menu) for the active document.

```
ActiveDocument.Compatibility(wdSuppressSpBfAfterPgBrk) = True
```

This example toggles the **Don't add automatic tab stop for hanging indent** option on or off.

```
ActiveDocument.Compatibility(wdNoTabHangIndent) = Not _  
    ActiveDocument.Compatibility(wdNoTabHangIndent)
```



↳ [Show All](#)

Compiled Property

-

True if the specified add-in is a [Word add-in library \(WLL\)](#). **False** if the add-in is a template. Read-only **Boolean**.

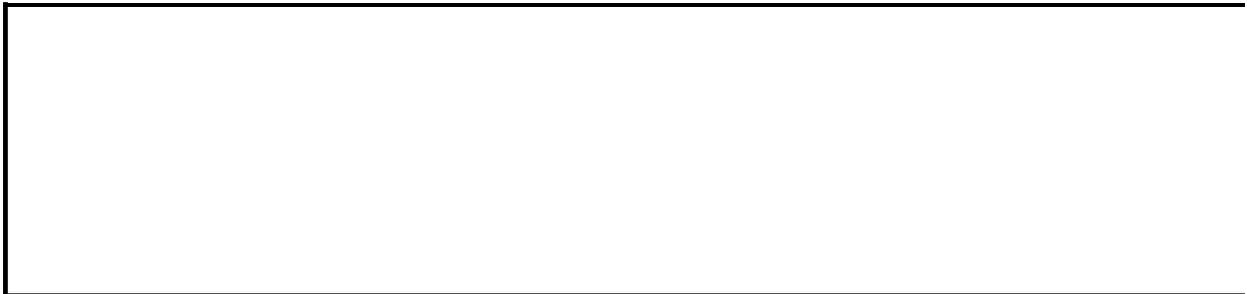
Example

This example determines how many WLLs are currently loaded.

```
count = 0
For Each aAddin in Addins
    If aAddin.Compiled = True And aAddin.Installed = True Then
        count = count + 1
    End If
Next aAddin
MsgBox Str(count) & " WLL's are loaded"
```

If the first add-in is a template, this example unloads the template and opens it.

```
If Addins(1).Compiled = False Then
    Addins(1).Installed = False
    Documents.Open FileName:=AddIns(1).Path _
        & Application.PathSeparator _
        & AddIns(1).Name
End If
```



ComposeStyle Property

-

Returns a [Style](#) object that represents the style used to compose new e-mail messages. Read-only.

Example

This example displays the name of the default style used to compose new e-mail messages.

```
MsgBox Application.EmailOptions.ComposeStyle.NameLocal
```

This example changes the font color of the default style used to compose new e-mail messages.

```
Application.EmailOptions.ComposeStyle.Font.Color = _  
    wdColorBrightGreen
```



ConfirmConversions Property

-

True if Word displays the **Convert File** dialog box before it opens or inserts a file that isn't a Word document or template. In the **Convert File** dialog box, the user chooses the format to convert the file from. Read/write **Boolean**.

Example

This example sets Word to display the **Convert File** dialog box whenever a file that isn't a Word document or template is opened.

```
Options.ConfirmConversions = True
```

This example returns the current status of the **Confirm conversion at Open** option on the **General** tab in the **Options** dialog box.

```
Dim blnConfirm As Boolean
```

```
blnConfirm= Options.ConfirmConversions
```



ConnectionString Property

-

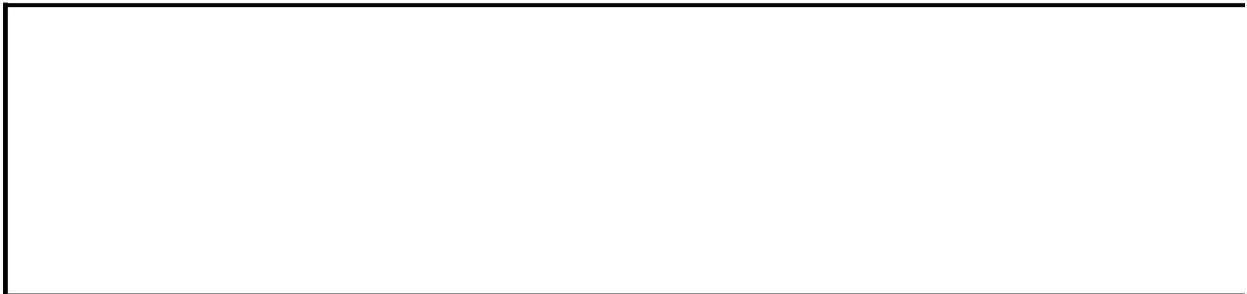
Returns the connection string for the specified mail merge data source. Read-only **String**.

Example

This example creates a new main document and attaches the Customers table from a Microsoft Access database named "Northwind.mdb." The connection string is displayed in a message box.

```
Dim docNew As Document
Set docNew = Documents.Add

With docNew.MailMerge
    .MainDocumentType = wdFormLetters
    .OpenDataSource _
        Name:="C:\Program Files\Microsoft Office\Office" & _
        "\Samples\Northwind.mdb", _
        LinkToSource:=True, AddToRecentFiles:=False, _
        Connection:="TABLE Customers"
    MsgBox .DataSource.ConnectionString
End With
```



ConsecutiveHyphensLimit Property

-
Returns or sets the maximum number of consecutive lines that can end with hyphens. Read/write. **Long**.

Note If this property is set to 0 (zero), any number of consecutive lines can end with hyphens.

Example

This example enables automatic hyphenation for MyReport.doc and limits the number of consecutive lines that can end with hyphens to two.

```
With Documents("MyReport.doc")  
    .AutoHyphenation = True  
    .ConsecutiveHyphensLimit = 2  
End With
```

This example sets no limit on the number of consecutive lines that can end with hyphens.

```
ActiveDocument.ConsecutiveHyphensLimit = 0
```



Container Property

-

Returns the object that represents the container application for the specified OLE object. Read-only.

Remarks

This property provides access to the specified document's container application if the document is embedded in another application as an OLE object.

The **Container** property also provides a pathway into the object model of the container application if a Word document is opened as an ActiveX document — for example, when a Word document is opened in Microsoft Office Binder or Internet Explorer.

Example

This example displays the name of the container application for the first shape in the active document. For the example to work, this shape must be an OLE object.

```
Msgbox ActiveDocument.Shapes(1).OLEFormat.Object.Container.Name
```



ContainingRange Property

-

Returns a [Range](#) object that represents the entire story in a series of shapes with linked text frames that the specified text frame belongs to. Read-only.

Example

This example checks the spelling in TextBox 1 and any other text in text frames that are linked to TextBox 1.

```
Dim rngStory As Range
```

```
Set rngStory = ActiveDocument.Shapes("TextBox 1") _  
    .TextFrame.ContainingRange
```

```
rngStory.CheckSpelling
```



↳ [Show All](#)

Content Property

-

Returns a **Range** object that represents the main document [story](#). Read-only.

Remarks

The following two statements are equivalent:

```
Set mainStory = ActiveDocument.Content
```

```
Set mainStory = ActiveDocument.StoryRanges(wdMainTextStory)
```

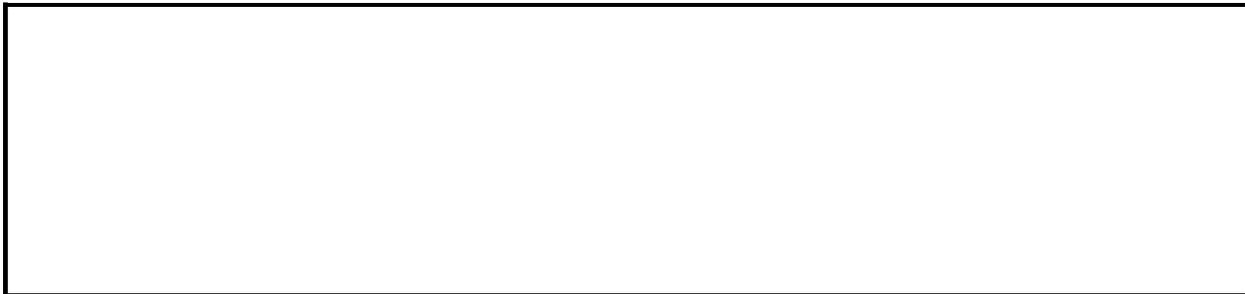
Example

This example changes the font and font size of the text in the active document to Arial 10 point.

```
Set myRange = ActiveDocument.Content
With myRange.Font
    .Name = "Arial"
    .Size = 10
End With
```

This example inserts text at the end of the document named "Changes.doc." The **For Each...Next** statement is used to determine whether the document is open.

```
For Each aDocument In Documents
    If InStr(LCase$(aDocument.Name), "changes.doc") Then
        Set myRange = Documents("Changes.doc").Content
        myRange.InsertAfter "the end."
    End If
Next aDocument
```



Context Property

-

Returns an object that represents the storage location of the specified key binding. This property can return a **Document**, **Template**, or **Application** object. Read-only.

Note Built-in key assignments (for example, CTRL+I for **Italic**) return the **Application** object as the context. Any key bindings you add will return a **Document** or **Template** object, depending on the customization context in effect when the **KeyBinding** object was added.

Example

This example displays the name of the document or template where the macro named "Macro1" is stored.

```
Sub TestContext1()  
    Dim kbMacro1 As KeysBoundTo  
  
    Set kbMacro1 = KeysBoundTo(KeyCategory:=wdKeyCategoryMacro, _  
        Command:="Macro1")  
    MsgBox kbMacro1.Context.Name  
End Sub
```

This example adds the F2 key to the **Italic** command and then uses the **For Each...Next** loop to display the keys assigned to the **Italic** command along with the context.

```
Dim kbLoop As KeyBinding  
  
CustomizationContext = NormalTemplate  
KeyBindings.Add KeyCategory:=wdKeyCategoryCommand, _  
    Command:="Italic", KeyCode:=wdKeyF2  
For Each kbLoop In _  
    KeysBoundTo(KeyCategory:=wdKeyCategoryCommand, _  
        Command:="Italic")  
    MsgBox kbLoop.KeyString & vbCr & kbLoop.Context.Name  
Next kbLoop
```



ContinuationNotice Property

-

Returns a **Range** object that represents the footnote or endnote continuation notice. Read-only.

Example

This example replaces the current footnote continuation notice with the text "Continued...".

```
With ActiveDocument.Footnotes.ContinuationNotice  
    .Delete  
    .InsertBefore "Continued..."  
End With
```



ContinuationSeparator Property

-

Returns a **Range** object that represents the footnote or endnote continuation separator. Read-only.

Example

This example replaces the current endnote continuation separator with a series of underscore characters.

```
With ActiveDocument.Endnotes.ContinuationSeparator  
    .Delete  
    .InsertBefore "____"  
End With
```



Contrast Property

-

Returns or sets 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 **Single**.

Example

This example sets the contrast for the first shape on the active document. The first shape must be either a picture or an OLE object.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
docActive.Shapes(1).PictureFormat.Contrast = 0.8
```



ConvertHighAnsiToFarEast Property

-
True if Microsoft Word converts text that is associated with an East Asian font to the appropriate font when it opens a document. Read/write **Boolean**.

expression.**ConvertHighAnsiToFarEast**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets Microsoft Word to convert text that is associated with an East Asian font to the appropriate font when it opens a document.

```
Options.ConvertHighAnsiToFarEast = True
```



ConvertMacWordChevrons Property

-

Controls whether text enclosed in chevron characters (« ») is converted to merge fields. Read/write **Long**. [WdChevronConvertRule](#)

Can be one of the following **WdChevronConvertRule** constants.

Constant	Description
wdAlwaysConvert	The converter attempts to convert text enclosed in chevrons (« ») to mail merge fields.
wdNeverConvert	The converter passes the text through without attempting any interpretation.
wdAskToConvert, wdAskToNotConvert	The converter prompts the user to convert or not convert chevrons when a Word for the Macintosh document is opened.

Remarks

Word for the Macintosh version 4.0 and 5.x documents use chevron characters to denote mail merge fields.

Example

This example sets the **ConvertMacWordChevrons** property to convert the text enclosed in chevrons to mail merge fields, and then it opens the document named "Mac Word Document."

```
FileConverters.ConvertMacWordChevrons = wdAlwaysConvert  
Documents.Open FileName:="C:\Documents\Mac Word Document"
```



CorrectCapsLock Property

-

True if Word automatically corrects instances in which you use the CAPS LOCK key inadvertently as you type. Read/write **Boolean**.

Example

This example determines whether Word is set to automatically correct CAPS LOCK key errors.

```
If AutoCorrect.CorrectCapsLock = True Then
    MsgBox "Correct CAPS LOCK is active."
Else
    MsgBox "Correct CAPS LOCK is not active."
End If
```



CorrectDays Property

-
True if Word automatically capitalizes the first letter of days of the week.
Read/write **Boolean**.

Example

This example sets Word to automatically capitalize the first letter of days of the week.

```
AutoCorrect.CorrectDays = True
```

This example toggles the value of the **CorrectDays** property.

```
AutoCorrect.CorrectDays = Not AutoCorrect.CorrectDays
```



CorrectHangulAndAlphabet Property

True if Microsoft Word automatically applies the correct font to Latin words typed in the middle of Hangul text or vice versa. Read/write **Boolean**.

expression.**CorrectHangulAndAlphabet**

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets Microsoft Word to automatically apply the correct font to Latin words typed in the middle of Hangul text or vice versa.

`AutoCorrect.CorrectHangulAndAlphabet = True`



CorrectHangulEndings Property

-
True if Microsoft Word automatically corrects Hangul endings when replacing Hangul text. Read/write **Boolean**.

expression.**CorrectHangulEndings**

expression Required. An expression that returns a [Find](#) object.

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Microsoft Word to automatically correct Hangul endings when replacing Hangul text.

```
With Selection.Find
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .CorrectHangulEndings = True
End With
```



CorrectInitialCaps Property

-
True if Word automatically makes the second letter lowercase if the first two letters of a word are typed in uppercase. For example, "WOrd" is corrected to "Word." Read/write **Boolean**.

Example

This example sets Word to automatically correct errors in initial capitalization.

```
AutoCorrect.CorrectInitialCaps = True
```



CorrectKeyboardSetting Property

-
True if Microsoft Word automatically transposes words to their native alphabet if you type text in a language other than the current keyboard language.
Read/write **Boolean**.

expression.**CorrectKeyboardSetting**

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

The [CheckLanguage](#) property must be set to **True** in order to use the **CorrectKeyboardSetting** property.

For more information on using Word with multiple languages, see [Troubleshoot multilingual text and automatic language detection](#).

Example

This example displays a dialog box where the user can choose whether or not Word automatically transposes foreign words to their native alphabets.

```
x = MsgBox("Do you want Microsoft Word to tranpose " _  
    & "foreign words to their native alphabet?", _  
    vbYesNo)  
If x = vbYes Then  
    Application.CheckLanguage = True  
    AutoCorrect.CorrectKeyboardSetting = True  
    MsgBox "Automatic keyboard correction enabled!"  
End If
```



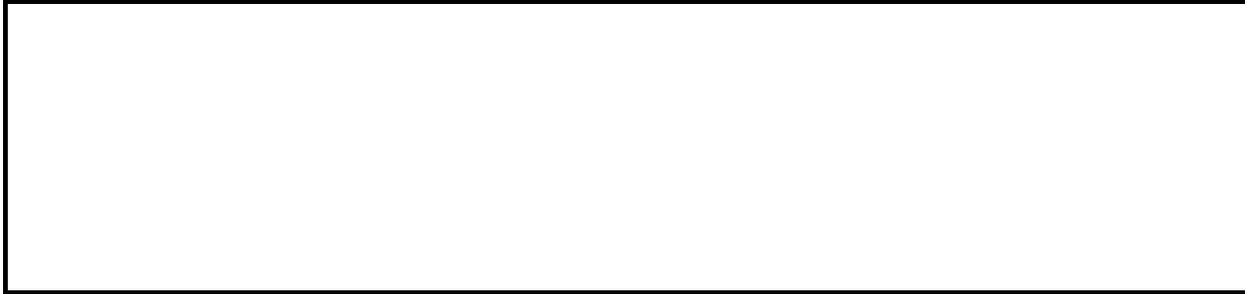
CorrectSentenceCaps Property

-
True if Word automatically capitalizes the first letter in each sentence.
Read/write **Boolean**.

Example

This example toggles the value of the **CorrectSentenceCaps** property.

```
AutoCorrect.CorrectSentenceCaps = Not _  
AutoCorrect.CorrectSentenceCaps
```



CorrectTableCells Property

True to automatically capitalize the first letter of table cells. Read/write **Boolean**.

expression.**CorrectTableCells**

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example disables automatic capitalization of the first letter typed within table cells.

```
Sub AutoCorrectFirstLetterOfTableCells()  
    Application.AutoCorrect.CorrectTableCells = False  
End Sub
```



Count Property

Returns the number of items in the specified collection. Read-only **Long**.

expression.**Count**

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

Example

This example displays the number of paragraphs in the active document.

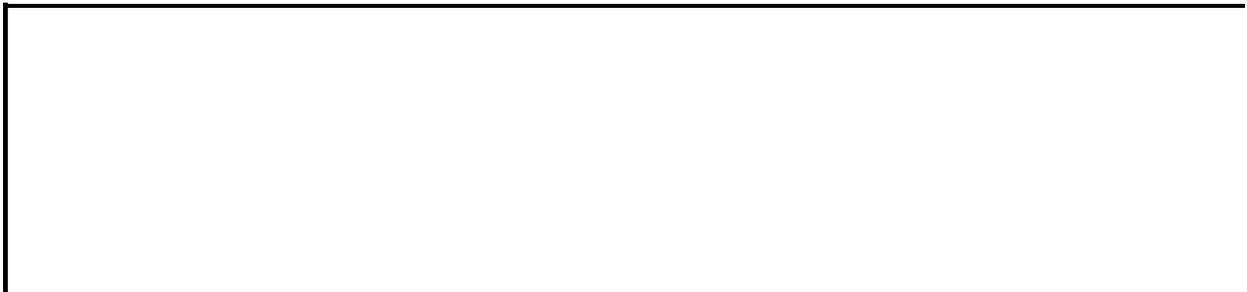
```
MsgBox "The active document contains " & _  
    ActiveDocument.Paragraphs.Count & "paragraphs."
```

This example displays the number of words in the selection.

```
If Selection.Words.Count >= 1 And _  
    Selection.Type <> wdSelectionIP Then  
    MsgBox "The selection contains " & Selection.Words.Count _  
        & " words."  
End If
```

This example uses the `aFields()` array to store the field codes in the active document.

```
fcount = ActiveDocument.Fields.Count  
If fcount >= 1 Then  
    ReDim aFields(fcount)  
    For Each aField In ActiveDocument.Fields  
        aFields(aField.Index) = aField.Code.Text  
    Next aField  
End If
```



CountBy Property

-

Returns or sets the numeric increment for line numbers. For example, if the **CountBy** property is set to 5, every fifth line will display the line number. Line numbers are only displayed in print layout view and print preview. Read/write **Long**.

Remarks

This property has no effect unless the **Active** property of the **LineNumbering** object is set to **True**.

Example

This example turns on line numbering for the active document. The line number is displayed on every fifth line and line numbering starts over for each new section.

```
With ActiveDocument.PageSetup.LineNumbering
    .Active = True
    .CountBy = 5
    .RestartMode = wdRestartSection
End With
```



Country Property

Returns the country/region designation of the system. Read-only [WdCountry](#).

WdCountry can be one of these WdCountry constants.

wdArgentina

wdCanada

wdChina

wdFinland

wdGermany

wdItaly

wdKorea

wdMexico

wdNorway

wdSpain

wdTaiwan

wdUS

wdBrazil

wdChile

wdDenmark

wdFrance

wdIceland

wdJapan

wdLatinAmerica

wdNetherlands

wdPeru

wdSweden

wdUK

wdVenezuela

expression.**Country**

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

Example

If the **Country** property returns **wdUS**, this example converts the top margin value from points to inches.

```
Dim sngMargin As Single
```

```
If System.Country = wdUS Then
```

```
    sngMargin = ActiveDocument.PageSetup.TopMargin
```

```
    MsgBox "Top margin is " & PointsToInches(sngMargin)
```

```
End If
```



CreateBackup Property

-
True if Word creates a backup copy each time a document is saved. Read/write
Boolean.

Remarks

The **CreateBackup** and **AllowFastSave** properties cannot be set to **True** concurrently.

Example

This example sets Word to automatically create a backup copy, and then it saves the active document.

```
Options.CreateBackup = True  
ActiveDocument.Save
```

This example returns the current status of the **Always create backup copy** option on the **Save** tab in the **Options** dialog box.

```
Dim blnBackup As Boolean  
  
blnBackup = Options.CreateBackup
```



Creator Property

Returns a 32-bit integer that indicates the application in which the specified object was created. For example, if the object was created in Microsoft Word, this property returns the hexadecimal number 4D535744, which represents the string "MSWD." This value can also be represented by the constant **wdCreatorCode**. Read-only **Long**.

expression.**Creator**

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

Remarks

The **Creator** property was primarily designed to be used on the Macintosh, where each application has a four-character creator code. For example, Microsoft Word has the creator code MSWD. For additional information about this property, consult the language reference Help included with Microsoft Office Macintosh Edition.

Example

This example displays a message about the creator of myObject.

```
Set myObject = ActiveDocument
If myObject.Creator = wdCreatorCode Then
    MsgBox "This is a Microsoft Word object"
Else
    MsgBox "This is not a Microsoft Word object"
End If
```



CropBottom Property

Returns or sets the number of points that are cropped off the bottom of the specified picture or OLE object. Read/write **Single**.

Note 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.

Example

This example crops 20 points off the bottom of shape three on the active document. For the example to work, shape three must be either a picture or an OLE object.

```
ActiveDocument.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 dblPercent As Double
Dim shapeCrop As Shape
Dim sngHeight As Single
Dim sngCrop As Single

dblPercent = Val(InputBox("What percentage do you want " _
    & "to crop off the bottom of this picture?"))

Set shapeCrop = _
    Selection.ShapeRange(1)

With shapeCrop.Duplicate
    .ScaleHeight 1, True
    sngHeight = .Height
    .Delete
End With

sngCrop = sngHeight * dblPercent / 100

shapeCrop.PictureFormat.CropBottom = sngCrop
```



CropLeft Property

Returns or sets the number of points that are cropped off the left side of the specified picture or OLE object. Read/write **Single**.

Note 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 side of your picture.

Example

This example crops 20 points off the left side of shape three on the active document. For the example to work, shape three must be either a picture or an OLE object.

```
ActiveDocument.Shapes(3).PictureFormat.CropLeft = 20
```

This example crops the percentage specified by the user off the left side 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 dblPercent As Double
Dim shapeCrop As Shape
Dim sngHeight As Single
Dim sngCrop As Single

dblPercent = Val(InputBox("What percentage do you want " _
    & "to crop off the left of this picture?"))

Set shapeCrop = _
    Selection.ShapeRange(1)

With shapeCrop.Duplicate
    .ScaleHeight 1, True
    sngHeight = .Height
    .Delete
End With

sngCrop = sngHeight * dblPercent / 100

shapeCrop.PictureFormat.Crop Left = sngCrop
```



CropRight Property

Returns or sets the number of points that are cropped off the right side of the specified picture or OLE object. Read/write **Single**.

Note 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 side of your picture.

Example

This example crops 20 points off the right side of shape three on the active document. For this example to work, shape three must be either a picture or an OLE object.

```
ActiveDocument.Shapes(3).PictureFormat.CropRight = 20
```

This example crops the percentage specified by the user off the right side 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 dblPercent As Double
Dim shapeCrop As Shape
Dim sngHeight As Single
Dim sngCrop As Single

dblPercent = Val(InputBox("What percentage do you want " _
    & "to crop off the right of this picture?"))

Set shapeCrop = _
    Selection.ShapeRange(1)

With shapeCrop.Duplicate
    .ScaleHeight 1, True
    sngHeight = .Height
    .Delete
End With

sngCrop = sngHeight * dblPercent / 100

shapeCrop.PictureFormat.Crop Right = sngCrop
```



CropTop Property

Returns or sets the number of points that are cropped off the top of the specified picture or OLE object. Read/write **Single**.

Note 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.

Example

This example crops 20 points off the top of shape three on the active document. For the example to work, shape three must be either a picture or an OLE object.

```
ActiveDocument.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 dblPercent As Double
Dim shapeCrop As Shape
Dim sngHeight As Single
Dim sngCrop As Single

dblPercent = Val(InputBox("What percentage do you want " _
    & "to crop off the top of this picture?"))

Set shapeCrop = _
    Selection.ShapeRange(1)

With shapeCrop.Duplicate
    .ScaleHeight 1, True
    sngHeight = .Height
    .Delete
End With

sngCrop = sngHeight * dblPercent / 100

shapeCrop.PictureFormat.CropTop = sngCrop
```



CtrlClickHyperlinkToOpen Property

True if Microsoft Word requires holding down the CTRL key while clicking to open a hyperlink. Read/write **Boolean**.

expression.**CtrlClickHyperlinkToOpen**

expression Required. An expression that returns an [Options](#) object.

Example

This example disables the option that requires holding down the CTRL key and clicking on hyperlinks to open them.

```
Sub ToggleHyperlinkOption()  
    Options.CtrlClickHyperlinkToOpen = False  
End Sub
```



CurrentEmailAuthor Property

-

Returns an [EmailAuthor](#) object that represents the author of the current e-mail message. Read-only.

Example

This example returns the name of the style associated with the current e-mail author.

```
MsgBox ActiveDocument.Email _  
    .CurrentEmailAuthor.Style.NameLocal
```



Cursor Property

-

Returns or sets the state (shape) of the pointer. Can be one of the following **WdCursorType** constants: **wdCursorIBeam**, **wdCursorNormal**, **wdCursorNorthwestArrow**, or **wdCursorWait**. Read/write **Long**.

Example

This example prints a message on the status bar and changes the pointer to a busy pointer.

```
Dim intWait As Integer

StatusBar = "Please wait..."

For intWait = 1 To 1000
    System.Cursor = wdCursorWait
Next intWait

StatusBar = "Task completed"
System.Cursor = wdCursorNormal
```



↳ [Show All](#)

CursorMovement Property

Returns or sets how the insertion point progresses within bidirectional text.
Read/write [WdCursorMovement](#).

WdCursorMovement can be one of these WdCursorMovement constants.

wdCursorMovementLogical Insertion point progresses according to the direction of the language Microsoft Word detects.

wdCursorMovementVisual Insertion point progresses to the next visually adjacent character.

expression.**CursorMovement**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the insertion point to progress to the next visually adjacent character as it moves through bidirectional text.

```
Options.CursorMovement = wdCursorMovementVisual
```



CustomDictionaries Property

Returns a [Dictionaries](#) object that represents the collection of active custom dictionaries. Active custom dictionaries are marked with a check in the **Custom Dictionaries** dialog box. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a new, blank custom dictionary to the collection. The path and file name of the new custom dictionary are then displayed in a message box.

```
Dim dicHome As Dictionary
Set dicHome = CustomDictionaries.Add(Filename:="Home.dic")
Msgbox dicHome.Path & Application.PathSeparator & dicHome.Name
```

This example removes all custom dictionaries so that no custom dictionaries are active. The custom dictionary files aren't deleted, though.

```
CustomDictionaries.ClearAll
```

This example displays the name of each custom dictionary in the collection.

```
Dim dicLoop As Dictionary

For Each dicLoop In CustomDictionaries
    MsgBox dicLoop.Name
Next dicLoop
```



CustomDocumentProperties Property

Returns a [DocumentProperties](#) collection that represents all the custom document properties for the specified document.

expression.**CustomDocumentProperties**

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

Remarks

Use the [BuiltInDocumentProperties](#) property to return the collection of built-in document properties.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example inserts a list of custom built-in properties at the end of the active document.

```
Set myRange = ActiveDocument.Content
myRange.Collapse Direction:=wdCollapseEnd
For Each prop In ActiveDocument.CustomDocumentProperties
    With myRange
        .InsertParagraphAfter
        .InsertAfter prop.Name & "= "
        .InsertAfter prop.Value
    End With
Next
```

This example adds a custom built-in property to Sales.doc.

```
thename = InputBox("Please type your name", "Name")
Documents("Sales.doc").CustomDocumentProperties.Add _
    Name:="YourName", LinkToContent:=False, Value:=thename, _
    Type:=msoPropertyTypeString
```



CustomizationContext Property

-

Returns or sets a [Template](#) or [Document](#) object that represents the template or document in which changes to menu bars, toolbars, and key bindings are stored. Corresponds to the value of the **Save in** box on the **Commands** tab in the **Customize** dialog box (**Tools** menu). Read/write.

Example

This example adds the ALT+CTRL+W key combination to the **FileClose** command. The keyboard customization is saved in the Normal template.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyControl, _
    wdKeyAlt, wdKeyW), _
    KeyCategory:=wdKeyCategoryCommand, Command:="FileClose"
```

This example adds the **File Versions** button to the **Standard** toolbar. The command bar customization is saved in the template attached to the active document.

```
CustomizationContext = ActiveDocument.AttachedTemplate
Application.CommandBars("Standard").Controls.Add _
    Type:=msoControlButton, _
    ID:=2522, Before:=8
```



CustomLabels Property

Returns a [CustomLabels](#) collection that represents the available custom mailing labels. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a new custom label named "AdminAddress" and then creates a page of mailing labels using a predefined return address.

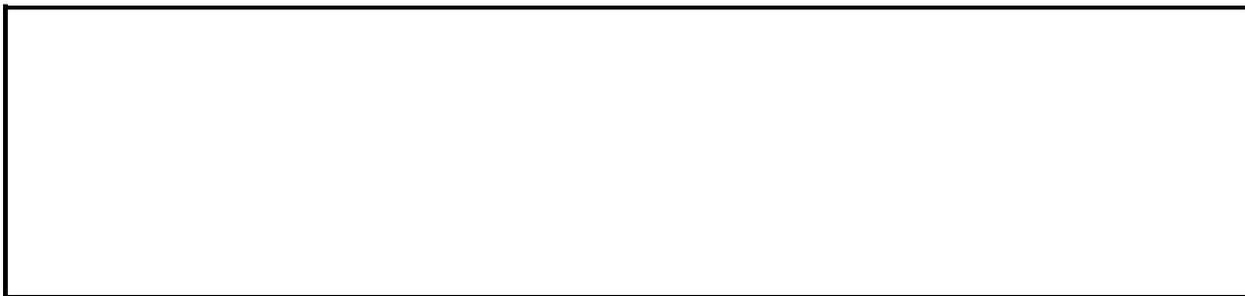
```
Dim strAddress As String
Dim labelNew As CustomLabel

strAddress = "Administration" & vbCrLf & "Mail Stop 22-16"

Set labelNew = Application.MailingLabel _
    .CustomLabels.Add(Name:="AdminAddress", DotMatrix:= False)

With labelNew
    .Height = InchesToPoints(0.5)
    .Width = InchesToPoints(1)
    .HorizontalPitch = InchesToPoints(2.06)
    .VerticalPitch = InchesToPoints(0.5)
    .NumberAcross = 4
    .NumberDown = 20
    .PageSize = wdCustomLabelLetter
    .SideMargin = InchesToPoints(0.28)
    .TopMargin = InchesToPoints(0.5)
End With

Application.MailingLabel.CreateNewDocument _
    Name:="AdminAddress", Address:=strAddress
```



CustomTab Property

-

True if the specified tab stop is a custom tab stop. Read-only **Boolean**.

Example

This example cycles through the collection of tab stops in the first paragraph in the active document, and left-aligns any custom tab stops that it finds.

```
Dim tsLoop As TabStop
```

```
For each tsLoop in ActiveDocument.Paragraphs(1).TabStops
```

```
    If tsLoop.CustomTab = True Then
```

```
        tsLoop.Alignment = wdAlignTabLeft
```

```
    End If
```

```
Next tsLoop
```



Cyan Property

-
Sets or returns a **Long** that represents the cyan component of a CMYK color.
Read-only.

expression.Cyan

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

Example

This example creates a new shape, then retrieves the four CMYK values from an existing shape in the active document, and then sets the CMYK fill color of the new 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.Shapes(1)  
    Set shpStar = ActiveDocument.Shapes.AddShape _  
        (Type:=msoShape5pointStar, Left:=200, _  
         Top:=100, Width:=150, Height:=150)  
  
    'Get current shapes CMYK colors  
    With shpHeart.Fill.ForeColor  
        lngCyan = .Cyan  
        lngMagenta = .Magenta  
        lngYellow = .Yellow  
        lngBlack = .Black  
    End With  
  
    'Sets new shape to current shapes CMYK colors  
    shpStar.Fill.ForeColor.SetCMYK _  
        Cyan:=lngCyan, Magenta:=lngMagenta, _  
        Yellow:=lngYellow, Black:=lngBlack  
End Sub
```



DashStyle Property

Returns or sets the dash style for the specified line. Read/write [MsoLineDashStyle](#).

MsoLineDashStyle can be one of these MsoLineDashStyle constants.

msoLineDashDot

msoLineDashStyleMixed

msoLineLongDashDot

msoLineSolid

msoLineDash

msoLineDashDotDot

msoLineLongDash

msoLineRoundDot

msoLineSquareDot

expression.DashStyle

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

Example

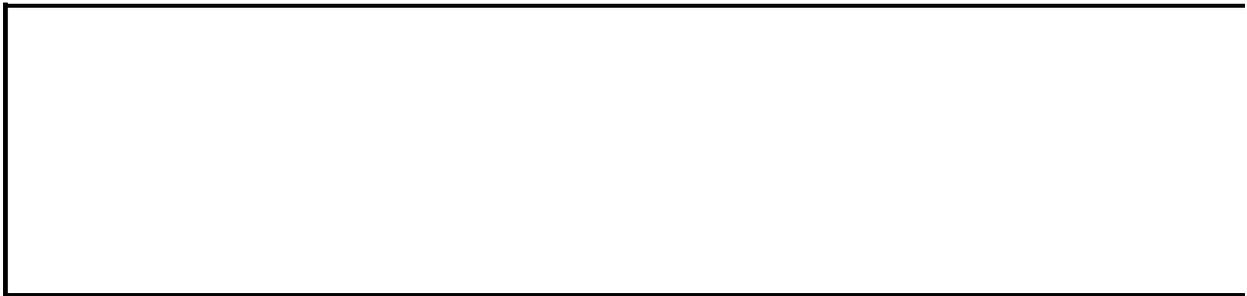
This example adds a blue dashed line to the active document.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(10, 10, 250, 250).Line  
    .DashStyle = msoLineDashDotDot  
    .ForeColor.RGB = RGB(50, 0, 128)
```

```
End With
```



Data Property

Returns or sets data in an ADDIN field. Read/write **String**.

Note The data is not visible in the field code or result; it is only accessible by returning the value of the **Data** property. If the field isn't an ADDIN field, this property will cause an error.

Example

This example inserts an ADDIN field at the insertion point in the active document and then sets the data for the field.

```
Dim fldTemp As Field
```

```
Selection.Collapse Direction:=wdCollapseStart
```

```
Set fldTemp = _
```

```
    ActiveDocument.Fields.Add(Range:=Selection.Range, _  
        Type:=wdFieldAddin)
```

```
fldTemp.Data = "Hidden information"
```



↳ [Show All](#)

DataFieldIndex Property

Returns or sets a **Long** that represents the corresponding field number in the mail merge data source to which a [mapped data field](#) maps. This property returns zero if the specified data field is not mapped to a mapped data field. Read/write.

expression.**DataFieldIndex**

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

Example

This example maps the PostalAddress1 field in the data source to the wdAddress1 mapped data field. This example assumes that the current document is a mail merge document.

```
Sub MapField()  
  With ThisDocument.MailMerge.DataSource  
    .MappedDataFields(wdAddress1).DataFieldIndex = _  
      .FieldNames("PostalAddress1").Index  
  End With  
End Sub
```



↳ [Show All](#)

DataFieldName Property

-

Sets or returns a **String** that represents the name of the field in the mail merge data source to which a [mapped data field](#) maps. A blank 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 a [MappedDataField](#) object.

Example

This example creates a tabbed list of the mapped data fields available in Word and the fields in the data source to which they are mapped. This example assumes that the current document is a mail merge document and that the data source fields have corresponding mapped data fields.

```
Sub MappedFields()  
    Dim intCount As Integer  
    Dim docCurrent As Document  
    Dim docNew As Document  
  
    On Error Resume Next  
  
    Set docCurrent = ThisDocument  
    Set docNew = Documents.Add  
  
    'Add leader tab to new document  
    docNew.Paragraphs.TabStops.Add _  
        Position:=InchesToPoints(3.5), _  
        Leader:=wdTabLeaderDots  
  
    With docCurrent.MailMerge.DataSource  
  
        'Insert heading paragraph for tabbed columns  
        docNew.Content.InsertAfter "Word Mapped Data Field" _  
            & vbTab & "Data Source Field"  
  
        Do  
            intCount = intCount + 1  
  
            'Insert Word mapped data field name and the  
            'corresponding data source field name  
            docNew.Content.InsertAfter .MappedDataFields( _  
                Index:=intCount).Name & vbTab & _  
                .MappedDataFields(Index:=intCount) _  
                .DataFieldName  
  
            'Insert paragraph  
            docNew.Content.InsertParagraphAfter  
        Loop Until intCount = .MappedDataFields.Count  
  
    End With
```

End Sub



DataFields Property

Returns a [MailMergeDataFields](#) collection that represents the fields in the specified mail merge data source. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

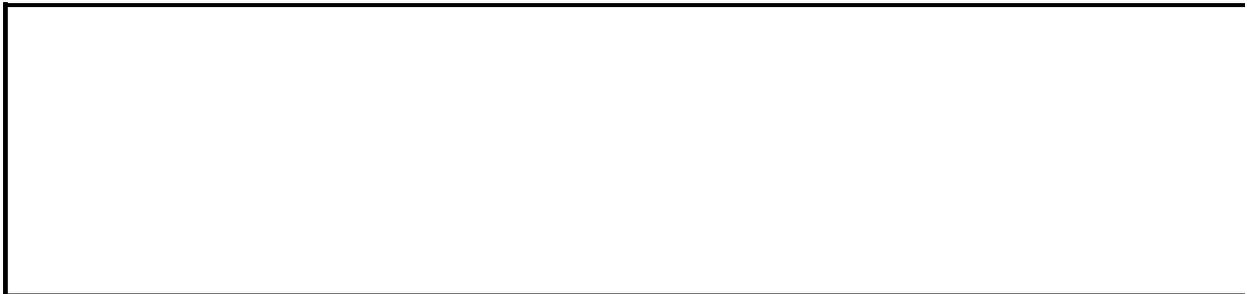
This example displays the name of each field in the data source attached to the active mail merge main document.

```
Dim mmdfTemp As MailMergeDataField

For Each mmdfTemp In _
    ActiveDocument.MailMerge.DataSource.DataFields
    MsgBox mmdfTemp.Name
Next mmdfTemp
```

This example displays the value of the LastName field from the first record in the data source attached to "Main.doc."

```
With Documents("Main.doc").MailMerge.DataSource
    .ActiveRecord = wdFirstRecord
    MsgBox DataFields("LastName").Value
End With
```



DataSource Property

-

Returns a [MailMergeDataSource](#) object that refers to the data source attached to a mail merge main document. Read-only.

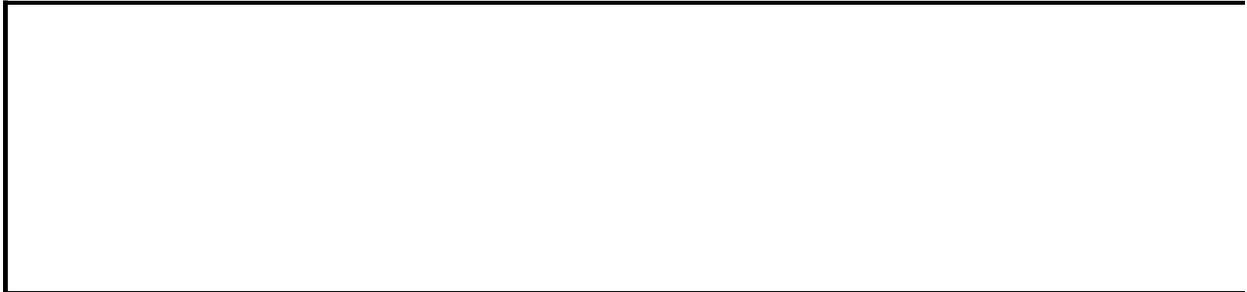
Example

This example displays the name of the data source attached to the active document.

```
If ActiveDocument.MailMerge.DataSource.Name <> "" Then _  
    MsgBox ActiveDocument.MailMerge.DataSource.Name
```

This example displays the next record from the data source attached to Main.doc.

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False  
With Documents("Main.doc").MailMerge  
    .ViewMailMergeFieldCodes = False  
    .DataSource.ActiveRecord = wdNextRecord  
End With
```



Date Property

-
Revision object: The date and time that the tracked change was made. Read-only **Date**.

Version object: The date and time that the document version was saved. Read-only **Date**.

Example

This example displays the date and time that the last version of the active document was saved.

```
Dim docActive As Document

Set docActive = ActiveDocument

If docActive.Path <> "" Then MsgBox _
    docActive.Versions(docActive.Versions.Count).Date
```

This example displays the date and time of the next tracked change found in the active document.

```
Dim revTemp As Revision

If ActiveDocument.Revisions.Count >= 1 Then
    Set revTemp = Selection.NextRevision
    If Not (revTemp Is Nothing) Then MsgBox revTemp.Date
End If
```



DateFormat Property

-

Returns or sets the date for a letter created by the Letter Wizard. Read/write **String**.

Example

This example displays the date from the letter that appears in the active document.

```
MsgBox ActiveDocument.GetLetterContent.DateFormat
```

This example creates a new **LetterContent** object, sets the date line to the current date, and then runs the Letter Wizard by using the **RunLetterWizard** method.

```
Dim lcNew As LetterContent
```

```
Set lcNew = New LetterContent
```

```
lcNew.DateFormat = Date$
```

```
ActiveDocument.RunLetterWizard LetterContent:=lcNew
```



↳ [Show All](#)

Default Property

▶ [Default property as it applies to the **CheckBox** object.](#)

Returns or sets the default check box value. **True** if the default value is checked. Read/write **Boolean**.

expression.**Default**

expression Required. An expression that returns one of the above objects.

▶ [Default property as it applies to the **DropDown** object.](#)

Returns or sets the default drop-down item. The first item in a drop-down form field is 1, the second item is 2, and so on. Read/write **Long**.

expression.**Default**

expression Required. An expression that returns one of the above objects.

▶ [Default property as it applies to the **TextInput** object.](#)

Returns or sets the text that represents the default text box contents. Read/write **String**.

expression.**Default**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **CheckBox** object.](#)

If the first form field in the active document is a check box, this example retrieves the default value.

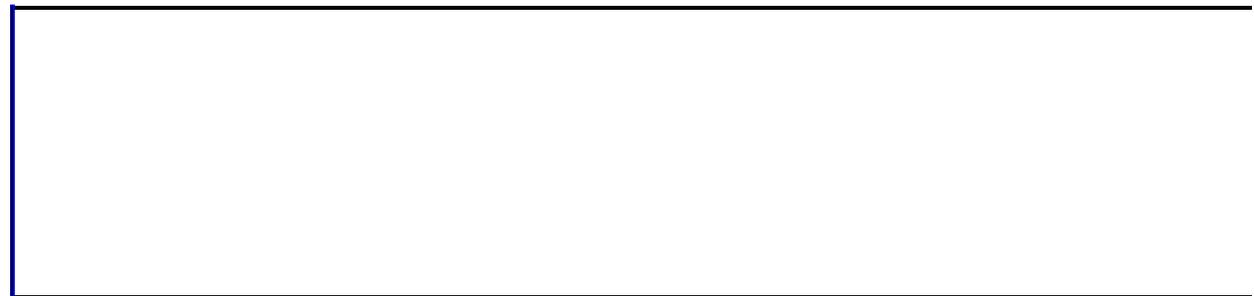
```
Dim blnDefault As Boolean
```

```
If ActiveDocument.FormFields(1).Type = wdFieldFormCheckBox Then  
    blnDefault = ActiveDocument.FormFields(1).CheckBox.Default  
End If
```

▶ [As it applies to the **DropDown** object.](#)

This example sets the default item for the drop-down form field named "Colors" in Sales.doc.

```
Documents("Sales.doc").FormFields("Colors").DropDown _  
    .Default = 2
```



▶ [As it applies to the **TextInput** object.](#)

This example sets the default text for the text form field named "Name."

```
ActiveDocument.FormFields("Name").TextInput.Default = _  
    "your name"
```

↳ [Show All](#)

DefaultBorderColor Property

Returns or sets the default 24-bit color to use for new [Border](#) objects. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**DefaultBorderColor**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets the default color for new borders to teal.

```
Options.DefaultBorderColor = wdColorTeal
```



DefaultBorderColorIndex Property

Returns or sets the default line color for borders. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**DefaultBorderColorIndex**

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

Note If the **Enable** property of the **Borders** object is set to **True**, the default

line width, line style, and line color for borders are used.

Example

This example changes the default line color and style for borders and then applies a border around the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Borders.Enable = True
With Options
    .DefaultBorderColorIndex = wdRed
    .DefaultBorderStyle = wdLineStyleDouble
End With
```



DefaultBorderStyle Property

Returns or sets the default border line style. Read/write [WdLineStyle](#).

WdLineStyle can be one of these WdLineStyle constants.

wdLineStyleDashDot

wdLineStyleDashDotDot

wdLineStyleDashDotStroked

wdLineStyleDashLargeGap

wdLineStyleDashSmallGap

wdLineStyleDot

wdLineStyleDouble

wdLineStyleDoubleWavy

wdLineStyleEmboss3D

wdLineStyleEngrave3D

wdLineStyleInset

wdLineStyleNone

wdLineStyleOutset

wdLineStyleSingle

wdLineStyleSingleWavy

wdLineStyleThickThinLargeGap

wdLineStyleThickThinMedGap

wdLineStyleThickThinSmallGap

wdLineStyleThinThickLargeGap

wdLineStyleThinThickMedGap

wdLineStyleThinThickSmallGap

wdLineStyleThinThickThinLargeGap

wdLineStyleThinThickThinMedGap

wdLineStyleThinThickThinSmallGap

wdLineStyleTriple

expression.**DefaultBorderStyle**

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

Example

This example sets the default line style to double.

```
Options.DefaultBorderStyle = wdLineStyleDouble
```

This example returns the current default line style.

```
Dim lngTemp As Long
```

```
lngTemp = Options.DefaultBorderStyle
```



DefaultBorderLineWidth Property

Returns or sets the default line width of borders. Read/write [WdLineWidth](#).

WdLineWidth can be one of these WdLineWidth constants.

wdLineWidth025pt

wdLineWidth050pt

wdLineWidth075pt

wdLineWidth100pt

wdLineWidth150pt

wdLineWidth225pt

wdLineWidth300pt

wdLineWidth450pt

wdLineWidth600pt

expression.**DefaultBorderLineWidth**

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

Note If the **Enable** property of the **Borders** object is set to **True**, the default line width and line style of borders are used.

Example

This example changes the default line width of borders and then adds a border around each paragraph in the selection.

```
Options.DefaultBorderLineWidth = wdLineWidth050pt  
Selection.Borders.Enable = True
```



DefaultEPostageApp Property

-

Sets or returns a **String** that represents the path and file name of the default electronic postage application. Read/write.

expression.**DefaultEPostageApp**

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

Example

This example specifies the path and file name for the default electronic postage application.

```
Sub DefaultEPostage()  
    Application.Options.DefaultEPostageApp = "C:\MyApp\EPostage.exe"  
End Sub
```



DefaultFaceUp Property

-

True if envelopes are fed face up by default. Read/write **Boolean**.

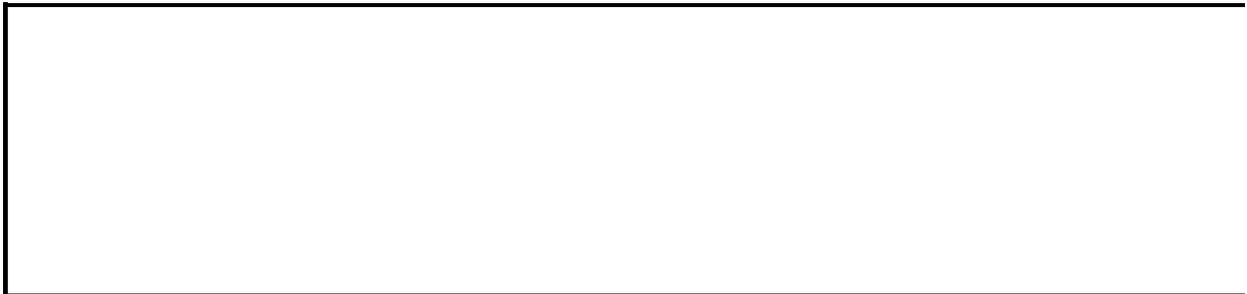
Example

This example sets envelopes to be fed face up by default. The **UpdateDocument** method updates the envelope in the active document.

```
With ActiveDocument.Envelope
    .DefaultFaceUp = True
    .DefaultOrientation = wdCenterPortrait
    .UpdateDocument
End With
```

This example displays a message telling the user how to feed the envelopes into the printer based on the default setting.

```
If ActiveDocument.Envelope.DefaultFaceUp = True Then
    MsgBox "Feed envelopes face up."
Else
    MsgBox "Feed envelopes face down."
End If
```



DefaultFilePath Property

Returns or sets default folders for items such as documents, templates, and graphics. Read/write **String**.

expression.DefaultFilePath(*Path*)

expression Required. An expression that returns an **Options** object.

Path Required [WdDefaultFilePath](#). The default folder to set.

WdDefaultFilePath can be one of these WdDefaultFilePath constants.

wdAutoRecoverPath

wdCurrentFolderPath

wdGraphicsFiltersPath

wdProgramPath

wdStartupPath

wdTempFilePath

wdToolsPath

wdUserOptionsPath

wdWorkgroupTemplatesPath

wdBorderArtPath

wdDocumentsPath

wdPicturesPath

wdProofingToolsPath

wdStyleGalleryPath

wdTextConvertersPath

wdTutorialPath

wdUserTemplatesPath

Remarks

The new setting takes effect immediately.

You can use an empty string ("") to remove the setting from the Windows registry.

Example

This example sets the default folder for Word documents.

```
Options.DefaultFilePath(wdDocumentsPath) = "C:\Documents"
```

This example returns the current default path for user templates (corresponds to the default path setting on the **File Locations** tab in the **Options** dialog box).

```
Dim strPath As String
```

```
strPath = Options.DefaultFilePath(wdUserTemplatesPath)
```



DefaultHeight Property

Returns or sets the default envelope height, in points. Read/write **Single**.

Note If you set either the **DefaultHeight** or **DefaultWidth** property, the envelope size is automatically changed to Custom Size in the **Envelope Options** dialog box (**Tools** menu). Use the [DefaultSize](#) property to set the default size to a predefined size.

Example

This example sets the default envelope size to 4.5 inches by 7.5 inches.

```
With ActiveDocument.Envelope  
    .DefaultHeight = InchesToPoints(4.5)  
    .DefaultWidth = InchesToPoints(7.5)  
End With
```



DefaultHighlightColorIndex Property

Returns or sets the color used to highlight text formatted with the **Highlight** button (**Formatting** toolbar). Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**DefaultHighlightColorIndex**

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

Example

This example sets the default highlight color to bright green. The new color doesn't apply to any previously highlighted text.

```
Options.DefaultHighlightColorIndex = wdBrightGreen
```

This example returns the current default highlight color index.

```
Dim lngTemp As Long
```

```
lngTemp = Options.DefaultHighlightColorIndex
```



DefaultLabelName Property

Returns or sets the name for the default mailing label. Read/write **String**.

Note To find the string for the specified built-in label, select the label in the **Label Options** dialog box (**Tools** menu, **Envelopes and Labels** dialog box, **Labels** tab, **Options** button). Then click **Details** and look at the **Label name** box, which contains the correct string to use for this property. To set a custom label as the default mailing label, use the label name that appears in the **Details** dialog box, or use the **Name** property with a **CustomLabel** object.

Remarks

Creating a new label document from a **CustomLabel** object automatically sets the **DefaultLabelName** property to the name of the **CustomLabel** object.

Example

This example returns the name of the current default mailing label.

```
Msgbox Application.MailingLabel.DefaultLabelName
```

This example sets the Avery Standard, 5160 Address label as the default mailing label.

```
Application.MailingLabel.DefaultLabelName = "5160"
```



DefaultLaserTray Property

Returns or sets the default paper tray that contains sheets of mailing labels.
Read/write [WdPaperTray](#).

WdPaperTray can be one of these WdPaperTray constants.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormatBin

wdPrinterTractorFeed

wdPrinterUpperBin

expression.**DefaultLaserTray**

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

Example

This example checks to determine whether the mailing label printer is set for feed labels manually, and then it displays a message on the status bar.

```
If Application.MailingLabel.DefaultLaserTray = _  
    wdPrinterManualEnvelopeFeed Then  
    StatusBar = "Printer set for feeding labels manually"  
Else  
    StatusBar = "Check printer paper tray setting"  
End If
```

This example sets the mailing-label paper tray to the upper bin.

```
Application.MailingLabel.DefaultLaserTray = wdPrinterUpperBin
```



DefaultLegalBlackline Property

-
True for Microsoft Word to compare and merge documents using the **Legal blackline** option in the **Compare and Merge Documents** dialog box.
Read/write **Boolean**.

expression.**DefaultLegalBlackline**

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

Remarks

For more information about the **Legal blackline** option, see [About comparing and merging documents](#) and [Compare documents with the Legal blackline option](#).

Example

This example enables Word's **Legal blackline** option for comparing and merging legal documents.

```
Sub CreateLegalBlackline()  
    Application.DefaultLegalBlackline = True  
End Sub
```



DefaultOmitReturnAddress Property

-
True if the return address is omitted from envelopes by default. Read/write **Boolean**.

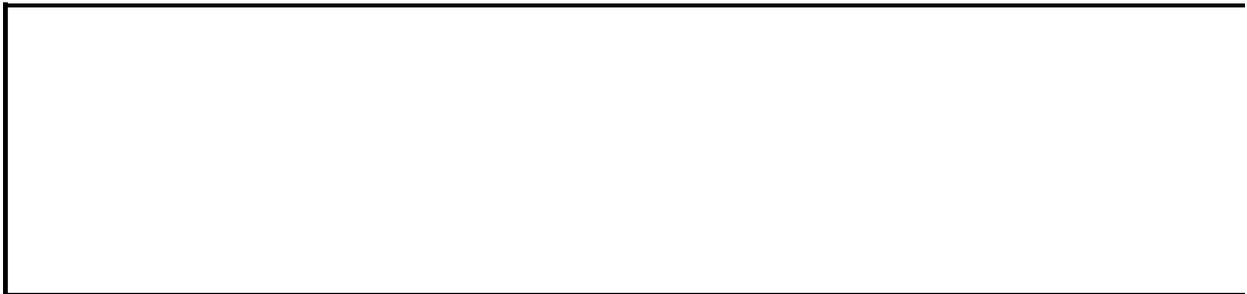
Example

This example omits return addresses from new envelopes by default.

```
ActiveDocument.Envelope.DefaultOmitReturnAddress = True
```

This example displays the return address status in a message box.

```
If ActiveDocument.Envelope.DefaultOmitReturnAddress = True Then  
    MsgBox "A return address is not included by default."  
Else  
    MsgBox "A return address is included by default."  
End If
```



DefaultOpenFormat Property

Returns or sets the default file converter used to open documents. Can be a number returned by the [OpenFormat](#) property, or one of the following [WdOpenFormat](#) constants.

WdOpenFormat can be one of these WdOpenFormat constants.

wdOpenFormatAllWord

wdOpenFormatAuto

wdOpenFormatDocument

wdOpenFormatEncodedText

wdOpenFormatRTF

wdOpenFormatTemplate

wdOpenFormatText

wdOpenFormatUnicodeText

wdOpenFormatWebPages

expression.DefaultOpenFormat

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

Note Use the **Format** argument with the [Open](#) method to specify a file converter when you're opening a single document.

Example

This example sets the default converter for opening documents to the Word document format and then opens Forecast.doc.

```
Options.DefaultOpenFormat = wdOpenFormatDocument  
Documents.Open FileName:="C:\Sales\Forecast.doc"
```

This example sets the default converter for opening documents to automatically determine the appropriate file converter to use when opening documents.

```
Options.DefaultOpenFormat = wdOpenFormatAuto
```

This example sets the default converter for opening documents to the WordPerfect 6.x format.

```
Options.DefaultOpenFormat = _  
    FileConverters("WordPerfect6x").OpenFormat
```



DefaultOrientation Property

Returns or sets the default orientation for feeding envelopes. Read/write [WdEnvelopeOrientation](#).

WdEnvelopeOrientation can be one of these WdEnvelopeOrientation constants.

wdCenterClockwise

wdCenterLandscape

wdCenterPortrait

wdLeftClockwise

wdLeftLandscape

wdLeftPortrait

wdRightClockwise

wdRightLandscape

wdRightPortrait

expression.**DefaultOrientation**

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

Example

This example sets envelopes to be fed face up, centered, and in portrait orientation.

```
With ActiveDocument.Envelope
    .DefaultFaceUp = True
    .DefaultOrientation = wdCenterPortrait
    MsgBox "Feed envelopes centered, face up, " _
        & "in portrait orientation"
End With
```



DefaultPrintBarCode Property

-
True if a POSTNET bar code is added to envelopes or mailing labels by default.
Read/write **Boolean**.

Note For U.S. mail only. For envelopes, this property must be set to **True** before the [DefaultPrintFIMA](#) property is set.

Example

This example sets the default envelope settings to include a bar code and a Facing Identification Mark (FIM-A).

```
With ActiveDocument.Envelope
    .DefaultPrintBarCode = True
    .DefaultPrintFIMA = True
End With
```

This example displays the bar code status in a message box.

```
If ActiveDocument.Envelope.DefaultPrintBarCode = False Then
    MsgBox "A bar code is not included by default"
Else
    MsgBox "A bar code is included by default"
End If
```



DefaultPrintFIMA Property

-
True to add a Facing Identification Mark (FIM-A) to envelopes by default.
Read/write **Boolean**.

Note For U.S. mail only. A FIM-A code is used to presort courtesy reply mail.
The [DefaultPrintBarCode](#) property must be set to **True** before this property is set.

Example

This example sets the default envelope settings to include a bar code and a Facing Identification Mark (FIM-A).

```
With ActiveDocument.Envelope  
    .DefaultPrintBarCode = True  
    .DefaultPrintFIMA = True  
End With
```



DefaultSaveFormat Property

-

Returns or sets the default format that will appear in the **Save as type** box in the **Save As** dialog box (**File** menu). Corresponds to the **Save Word files as** box on the **Save** tab in the **Options** dialog box (**Tools** menu). Read/write **String**.

Remarks

The string used with this property is the file converter class name. The class names for internal Word formats are listed in the following table.

Word format	File converter class name
Word Document	""
Document Template	"Dot"
Text Only	"Text"
Text Only with Line Breaks	"CRText"
MS-DOS Text	"8Text"
MS-DOS Text with Line Breaks	"8CRText"
Rich Text Format	"Rtf"
Unicode Text	"Unicode"

Use the [ClassName](#) property with a **FileConverter** object to determine the class name of an external file converter.

Example

This example sets the Word document format as the default save format.

```
Application.DefaultSaveFormat = ""
```

This example returns the current setting the **Save Word files as** box on the **Save** tab in the **Options** dialog box (**Tools** menu).

```
Msgbox Application.DefaultSaveFormat
```



DefaultSize Property

Returns or sets the default envelope size. Read/write **String**.

Note The string that's returned corresponds to the right-hand side of the string that appears in the **Envelope Size** box in the **Envelope Options** dialog box. If you set either the [DefaultHeight](#) or [DefaultWidth](#) property, the envelope size is automatically changed to Custom Size in the **Envelope Options** dialog box (**Tools** menu) and this property returns "Custom size."

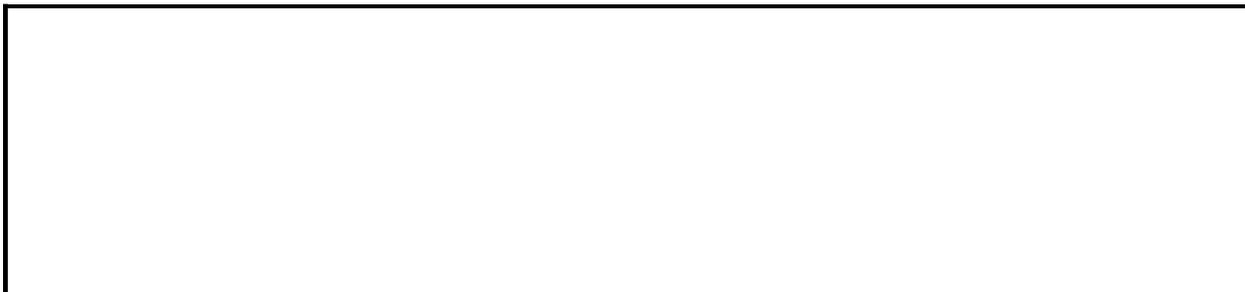
Example

This example sets the default envelope size to C4 (229 x 324 mm).

```
ActiveDocument.Envelope.DefaultSize = "C4"
```

This example asks the user whether or not they want to change the default envelope size to Size 10. If the answer is yes, the default size is changed accordingly. The **UpdateDocument** method changes the envelope size for the active document. If an envelope has not been added to the active document, a message box is displayed.

```
Sub exDefaultSize()  
  
    Dim intResponse As Integer  
  
    On Error GoTo errhandler  
    intResponse = MsgBox("Do you want to set the " _  
        & "default envelope to Size 10?", 4)  
    If intResponse = vbYes Then  
        With ActiveDocument.Envelope  
            .DefaultSize = "Size 10"  
            .UpdateDocument  
        End With  
    End If  
  
    Exit Sub  
  
errhandler:  
    If Err = 5852 Then _  
        MsgBox "An envelope isn't part of this document"  
End Sub
```



DefaultSorting Property

Returns or sets the sorting option for bookmark names displayed in the **Bookmark** dialog box (**Insert** menu). Read/write [WdBookmarkSortBy](#).

WdBookmarkSortBy can be one of these WdBookmarkSortBy constants.

wdSortByLocation

wdSortByName

expression.**DefaultSorting**

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

Remarks

This property doesn't affect the order of **Bookmark** objects in the **Bookmarks** collection.

Example

This example sorts bookmarks by location and then displays the **Bookmark** dialog box.

```
ActiveDocument.Bookmarks.DefaultSorting = wdSortByLocation  
Dialogs(wdDialogInsertBookmark).Show
```



DefaultTab Property

Returns or sets the active tab when the specified dialog box is displayed. Read/write [WdWordDialogTab](#).

WdWordDialogTab can be one of these WdWordDialogTab constants.

wdDialogEmailOptionsTabSignature

wdDialogFilePageSetupTabCharsLines

wdDialogFilePageSetupTabMargins

wdDialogFilePageSetupTabPaperSize

wdDialogFormatBordersAndShadingTabBorders

wdDialogFormatBordersAndShadingTabShading

wdDialogFormatBulletsAndNumberingTabNumbered

wdDialogFormatDrawingObjectTabColorsAndLines

wdDialogFormatDrawingObjectTabPicture

wdDialogFormatDrawingObjectTabSize

wdDialogFormatDrawingObjectTabWeb

wdDialogFormatFontTabAnimation

wdDialogFormatFontTabFont

wdDialogFormatParagraphTabTeisai

wdDialogInsertIndexAndTablesTabIndex

wdDialogInsertIndexAndTablesTabTableOfContents

wdDialogInsertSymbolTabSpecialCharacters

wdDialogLetterWizardTabLetterFormat

wdDialogLetterWizardTabRecipientInfo

wdDialogNoteOptionsTabAllEndnotes

wdDialogOrganizerTabAutoText

wdDialogOrganizerTabMacros

wdDialogTablePropertiesTabCell

wdDialogTablePropertiesTabRow

wdDialogEmailOptionsTabQuoting
wdDialogEmailOptionsTabStationary
wdDialogFilePageSetupTabLayout
wdDialogFilePageSetupTabPaper
wdDialogFilePageSetupTabPaperSource
wdDialogFormatBordersAndShadingTabPageBorder
wdDialogFormatBulletsAndNumberingTabBulleted
wdDialogFormatBulletsAndNumberingTabOutlineNumbered
wdDialogFormatDrawingObjectTabHR
wdDialogFormatDrawingObjectTabPosition
wdDialogFormatDrawingObjectTabTextbox
wdDialogFormatDrawingObjectTabWrapping
wdDialogFormatFontTabCharacterSpacing
wdDialogFormatParagraphTabIndentsAndSpacing
wdDialogFormatParagraphTabTextFlow
wdDialogInsertIndexAndTablesTabTableOfAuthorities
wdDialogInsertIndexAndTablesTabTableOfFigures
wdDialogInsertSymbolTabSymbols
wdDialogLetterWizardTabOtherElements
wdDialogLetterWizardTabSenderInfo
wdDialogNoteOptionsTabAllFootnotes
wdDialogOrganizerTabCommandBars
wdDialogOrganizerTabStyles
wdDialogTablePropertiesTabColumn
wdDialogTablePropertiesTabTable
wdDialogToolsAutoCorrectExceptionsTabFirstLetter
wdDialogToolsAutoCorrectExceptionsTabHangulAndAlphabet
wdDialogToolsAutoCorrectExceptionsTabIac
wdDialogToolsAutoCorrectExceptionsTabInitialCaps
wdDialogToolsAutoManagerTabAutoCorrect
wdDialogToolsAutoManagerTabAutoFormat
wdDialogToolsAutoManagerTabAutoFormatAsYouType
wdDialogToolsAutoManagerTabAutoText

wdDialogToolsAutoManagerTabTraits
wdDialogToolsEnvelopesAndLabelsTabEnvelopes
wdDialogToolsEnvelopesAndLabelsTabLabels
wdDialogToolsOptionsTabAcetate
wdDialogToolsOptionsTabBidi
wdDialogToolsOptionsTabCompatibility
wdDialogToolsOptionsTabEdit
wdDialogToolsOptionsTabFileLocations
wdDialogToolsOptionsTabFuzzy
wdDialogToolsOptionsTabGeneral
wdDialogToolsOptionsTabHangulHanjaConversion
wdDialogToolsOptionsTabPrint
wdDialogToolsOptionsTabProofread
wdDialogToolsOptionsTabSave
wdDialogToolsOptionsTabSecurity
wdDialogToolsOptionsTabTrackChanges
wdDialogToolsOptionsTabTypography
wdDialogToolsOptionsTabUserInfo
wdDialogToolsOptionsTabView
wdDialogWebOptionsBrowsers
wdDialogWebOptionsEncoding
wdDialogWebOptionsFiles
wdDialogWebOptionsFonts
wdDialogWebOptionsGeneral
wdDialogWebOptionsPictures

expression.**DefaultTab**

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

Example

This example displays the **Page Setup** dialog box with the **Paper Source** tab selected.

```
With Dialogs(wdDialogFilePageSetup)  
    .DefaultTab = wdDialogFilePageSetupTabPaperSource  
    .Show  
End With
```



DefaultTableSeparator Property

Returns or sets the single character used to separate text into cells when text is converted to a table. Read/write **String**.

Note The value of the **DefaultTableSeparator** property is used if the *Separator* argument is omitted from the [ConvertToTable](#) method.

Example

This example changes the default table separator character.

```
Application.DefaultTableSeparator = "%"
```



DefaultTableStyle Property

Returns a **Variant** that represents the table style that is applied to all newly created tables in a document. Read-only.

expression.**DefaultTableStyle**

expression Required. An expression that returns a [Document](#) object.

Example

This example checks to see if the default table style used in the active document is named "Table Normal" and, if it is, changes the default table style to "TableStyle1." This example assumes that you have a table style named "TableStyle1."

```
Sub TableDefaultStyle()  
  With ActiveDocument  
    If .DefaultTableStyle = "Table Normal" Then  
      .SetDefaultTableStyle _  
        Style:="TableStyle1", SetInTemplate:=True  
    End If  
  End With  
End Sub
```



DefaultTabStop Property

-
Returns or sets the interval (in points) between the default tab stops in the specified document. Read/write **Single**.

Example

This example sets the default tab stops in the active document to 1 inch. The **InchesToPoints** method is used to convert inches to points.

```
ActiveDocument.DefaultTabStop = InchesToPoints(1)
```



DefaultTargetFrame Property

Returns or sets a **String** indicating the browser frame in which to display a Web page reached through a hyperlink. Read/write.

expression.**DefaultTargetFrame**

expression Required. An expression that returns a [Document](#) object.

Remarks

While the **DefaultTargetFrame** property can use any user-defined string, it has the following predefined strings: "_top", "_blank", "_parent", and "_self".

Example

This example sets Microsoft Word to open a new blank browser window when a user clicks on hyperlinks in the active document.

```
Sub DefaultFrame()  
    ActiveDocument.DefaultTargetFrame = "_blank"  
End Sub
```



↳ [Show All](#)

DefaultTextEncoding Property

Returns or sets an [MsoEncoding](#) constant representing the code page, or character set, that Microsoft Word uses for all documents saved as encoded text files. Read/write.

MsoEncoding can be one of these MsoEncoding constants.

msoEncodingArabic

msoEncodingArabicASMO

msoEncodingArabicAutoDetect Not used with this property.

msoEncodingArabicTransparentASMO

msoEncodingAutoDetect Not used with this property.

msoEncodingBaltic

msoEncodingCentralEuropean

msoEncodingCyrillic

msoEncodingCyrillicAutoDetect Not used with this property.

msoEncodingEBCDICArabic

msoEncodingEBCDICDenmarkNorway

msoEncodingEBCDICFinlandSweden

msoEncodingEBCDICFrance

msoEncodingEBCDICGermany

msoEncodingEBCDICGreek

msoEncodingEBCDICGreekModern

msoEncodingEBCDICHebrew

msoEncodingEBCDICIcelandic

msoEncodingEBCDICInternational

msoEncodingEBCDICItaly

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDICKoreanExtended
msoEncodingEBCDICKoreanExtendedAndKorean
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICMultilingualROECELatin2
msoEncodingEBCDICRussian
msoEncodingEBCDICSerbianBulgarian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICThai
msoEncodingEBCDICTurkish
msoEncodingEBCDICTurkishLatin5
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanada
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEBCDICUSCanadaAndTraditionalChinese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect Not used with this property.
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
msoEncodingISO2022CNSimplifiedChinese
msoEncodingISO2022CNTraditionalChinese
msoEncodingISO2022JPJISX02011989
msoEncodingISO2022JPJISX02021984
msoEncodingISO2022JPNoHalfwidthKatakana

msoEncodingISO2022KR
msoEncodingISO6937NonSpacingAccent
msoEncodingISO885915Latin9
msoEncodingISO88591Latin1
msoEncodingISO88592CentralEurope
msoEncodingISO88593Latin3
msoEncodingISO88594Baltic
msoEncodingISO88595Cyrillic
msoEncodingISO88596Arabic
msoEncodingISO88597Greek
msoEncodingISO88598Hebrew
msoEncodingISO88599Turkish
msoEncodingJapaneseAutoDetect Not used with this property.
msoEncodingJapaneseShiftJIS
msoEncodingKOI8R
msoEncodingKOI8U
msoEncodingKorean
msoEncodingKoreanAutoDetect Not used with this property.
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 Not used with this property.
msoEncodingSimplifiedChineseGBK
msoEncodingT61
msoEncodingTaiwanCNS
msoEncodingTaiwanEten
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTCA
msoEncodingTaiwanTeleText
msoEncodingTaiwanWang
msoEncodingThai
msoEncodingTraditionalChineseAutoDetect Not used with this property.
msoEncodingTraditionalChineseBig5
msoEncodingTurkish
msoEncodingUnicodeBigEndian
msoEncodingUnicodeLittleEndian
msoEncodingUSASCII
msoEncodingUTF7

msoEncodingUTF8
msoEncodingVietnamese
msoEncodingWestern

expression.**DefaultTextEncoding**

expression Required. An expression that returns an [Options](#) object.

Remarks

Use the [TextEncoding](#) property to set the encoding for an individual document.
To set encoding for HTML documents, use the [Encoding](#) property.

Example

This example sets the global text encoding to the Western code page. This means that Word will save all encoded text files using the Western code page.

```
Sub DefaultEncode()  
    Application.Options.DefaultTextEncoding = msoEncodingWestern  
End Sub
```



DefaultTray Property

-
Returns or sets the default tray your printer uses to print documents. Read/write **String**.

Remarks

When setting this property, you must specify a string found in the **Default tray** box on the **Print** tab in the **Options** dialog box. You can use the [DefaultTrayID](#) property and specify a **WdPaperTray** constant to set this same option.

Example

This example sets Word up to use the lower print tray.

```
Options.DefaultTray = "Lower tray"
```

This example returns the string found in the **Default tray** box on the **Print** tab in the **Options** dialog box.

```
Msgbox Options.DefaultTray
```



DefaultTrayID Property

Returns or sets the default tray your printer uses to print documents. Read/write [WdPaperTray](#).

WdPaperTray can be one of these WdPaperTray constants.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormalBin

wdPrinterTractorFeed

wdPrinterUpperBin

Remarks

You can use the [DefaultTray](#) property with a string from the **Default tray** box on the **Print** tab in the **Options** dialog box to set this same option.

Example

This example sets Word to use the upper print tray, and then it prints the active document.

```
Options.DefaultTrayID = wdPrinterUpperBin  
ActiveDocument.PrintOut
```

This example returns the current setting of the **Default tray** option on the **Print** tab in the **Options** dialog box.

```
Dim lngTray As Long  
  
lngTray = Options.DefaultTrayID
```



DefaultWidth Property

Returns or sets the default envelope width, in points. Read/write **Single**

Note If you set the **DefaultHeight** or **DefaultWidth** property, the envelope size is automatically changed to Custom Size in the **Envelope Options** dialog box (**Tools** menu). Use the [DefaultSize](#) property to set the default size to a predefined size.

Example

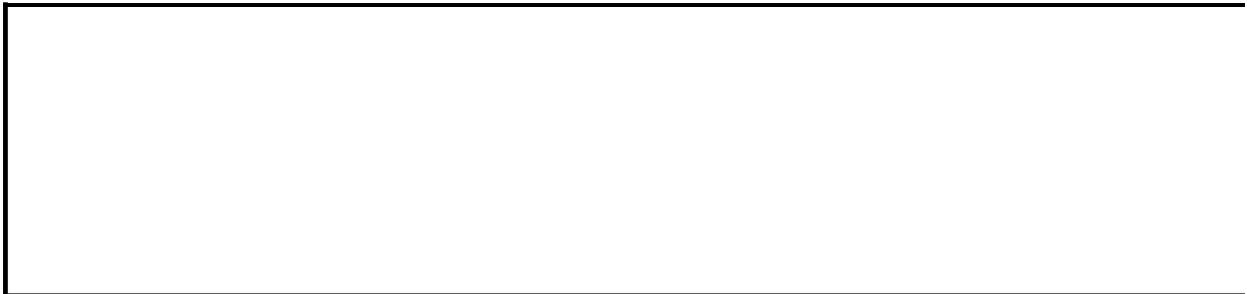
This example sets the default custom envelope width and height and adds an envelope to the active document.

```
Dim strAddress As String
Dim strReturn As String

strAddress = "Tim O' Brien " & vbCr & "123 Skye St." _
    & vbCr & "Bellevue, WA 98004"
strReturn = "Dave Edson" & vbCr & "123 West Main" _
    & vbCr & "Seattle, WA 98004"

With ActiveDocument.Envelope
    .DefaultWidth = InchesToPoints(9)
    .DefaultHeight = InchesToPoints(3.85)
End With

ActiveDocument.Envelope.Insert _
    Address:=strAddress, ReturnAddress:=strReturn
```



DefaultWritingStyle Property

-

Returns or sets the default writing style used by the grammar checker for the specified language. The name of the writing style is the localized name for the specified language. Read/write **String**.

Remarks

This property controls the global setting for the writing style. When setting this property, you must use the exact name found in the **Writing style** box on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

The [ActiveWritingStyle](#) property sets the writing style for each language in a document. The **ActiveWritingStyle** setting overrides the **DefaultWritingStyle** setting.

Example

This example returns the default writing style in a message box.

```
Dim lngLanguage As Long
```

```
lngLanguage = Selection.LanguageID  
Msgbox Languages(lngLanguage).DefaultWritingStyle
```

This example sets the writing style for U.S. English to Casual, and then it checks spelling and grammar in the active document.

```
Languages(wdEnglishUS).DefaultWritingStyle = "Casual"  
ActiveDocument.CheckGrammar
```



DeletedTextColor Property

Returns or sets the color of text that is deleted while change tracking is enabled. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.DeletedTextColor

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

Remarks

If the **DeletedTextColor** property is set to **wdByAuthor**, Word automatically assigns a unique color to each of the first eight authors who revise a document.

Example

This example sets the color of deleted text to bright green.

```
Options.DeletedTextColor = wdBrightGreen
```

This example returns the current status of the **Color** option under **Deleted Text** on the **Track Changes** tab in the **Options** dialog box.

```
Dim lngTemp As Long
```

```
lngTemp = Options.DeletedTextColor
```



DeletedTextMark Property

Returns or sets the format of text that is deleted while change tracking is enabled. Read/write [WdDeletedTextMark](#).

WdDeletedTextMark can be one of these WdDeletedTextMark constants.

wdDeletedTextMarkCaret

wdDeletedTextMarkPound

wdDeletedTextMarkHidden

wdDeletedTextMarkStrikeThrough

expression.DeletedTextMark

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

Example

This example applies strikethrough formatting to deleted text.

```
Options.DeletedTextMark = wdDeletedTextMarkStrikeThrough
```

This example returns the current status of the **Mark** option under **Deleted Text** on the **Track Changes** tab in the **Options** dialog box.

```
Dim lngTemp As Long
```

```
lngTemp = Options.DeletedTextMark
```



Delivery Property

Returns or sets the delivery method used for routing the document. Read/write [WdRoutingSlipDelivery](#). Read/write **Long** before routing starts; read-only **Long** while routing is in progress.

WdRoutingSlipDelivery can be one of these WdRoutingSlipDelivery constants.

wdAllAtOnce

wdOneAfterAnother

expression.**Delivery**

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

Example

This example routes the document named "Status.doc" to two recipients, one after the other.

```
Documents("Status.doc").HasRoutingSlip = True
With Documents("Status.doc").RoutingSlip
    .Subject = "Status Doc"
    .AddRecipient Recipient:="Don Funk"
    .AddRecipient Recipient:="Eric Maffei"
    .Delivery = wdOneAfterAnother
End With
Documents("Status.doc").Route
```



Depth Property

-

Returns or sets the depth of the shape's extrusion. Can be a value from – 600 through 9600 (positive values produce an extrusion whose front face is the original shape; negative values produce an extrusion whose back face is the original shape). Read/write **Single**.

Example

This example adds an oval to the active document and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Dim docActive As Document
Dim shapeNew As Shape

Set docActive = ActiveDocument
Set shapeNew = docActive.Shapes.AddShape(msoShapeOval, _
    90, 90, 90, 40)

With shapeNew.ThreeD
    .Visible = True
    .Depth = 50
    ' RGB value for purple
    .ExtrusionColor.RGB = RGB(255, 100, 255)
End With
```



Description Property

-

Returns the description of the specified style. For example, a typical description for the Heading 2 style might be "Normal + Font: Arial, 12 pt, Bold, Italic, Space Before 12 pt After 3 pt, KeepWithNext, Level 2." Read-only **String**.

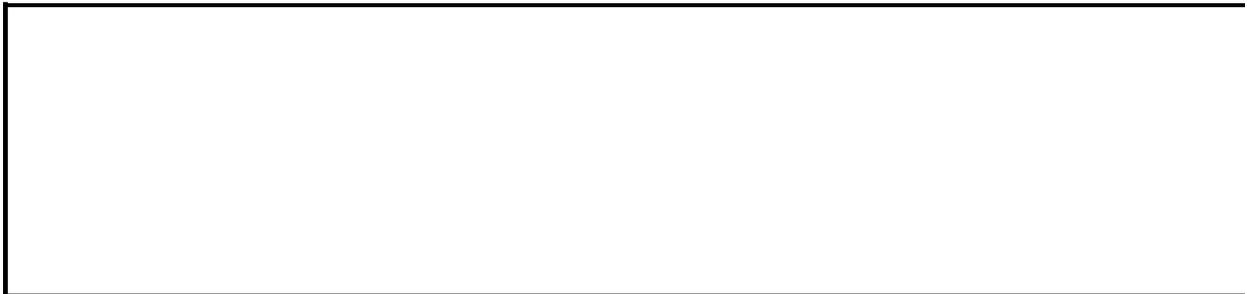
Example

This example creates a new document and inserts a tab-delimited list of the active document's styles and their descriptions.

```
Dim docActive As Document
Dim docNew As Document
Dim styleLoop As Style

Set docActive = ActiveDocument
Set docNew = Documents.Add

For Each styleLoop In docActive.Styles
    With docNew.Range
        .InsertAfter Text:=styleLoop.NameLocal & Chr(9) _
            & styleLoop.Description
        .InsertParagraphAfter
    End With
Next styleLoop
```



Destination Property

Returns or sets the destination of the mail merge results. Read/write [WdMailMergeDestination](#).

WdMailMergeDestination can be one of these WdMailMergeDestination constants.

wdSendToFax

wdSendToPrinter

wdSendToEmail

wdSendToNewDocument

expression.**Destination**

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

Example

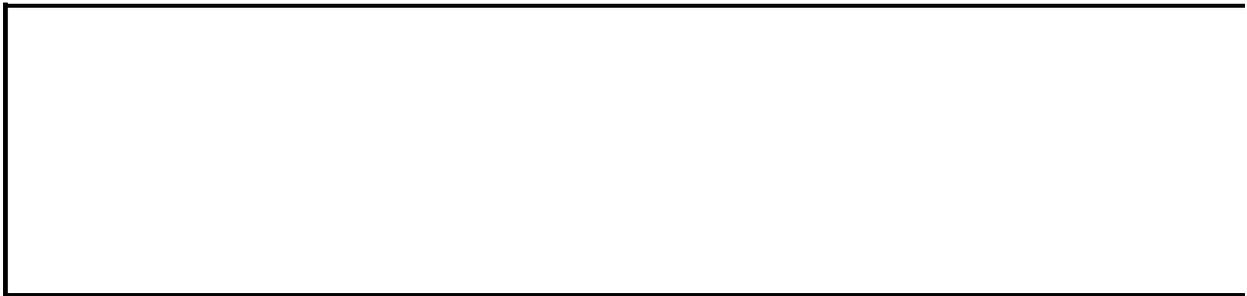
This example sends the results of a mail merge operation to a new document.

```
Dim mmTemp As MailMerge
```

```
Set mmTemp = ActiveDocument.MailMerge
```

```
If mmTemp.State = wdMainAndDataSource Then  
    mmTemp.Destination = wdSendToNewDocument  
    mmTemp.Execute
```

```
End If
```



↳ [Show All](#)

DiacriticColor Property

Returns or sets the 24-bit color to be used for diacritics for the specified [Font](#) object. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**DiacriticColor**

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

Remarks

The value of the [UseDiffDiacColor](#) property must be **True** in order to use this property.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the color for diacritics to blue in the current selection.

```
If Options.UseDiffDiacColor = True Then _  
    Selection.Font.DiacriticColor = wdColorBlue
```



↳ [Show All](#)

DiacriticColorVal Property

Returns or sets the 24-bit color to be used for diacritics in a right-to-left language document. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**DiacriticColorVal**

expression Required. An expression that returns an [Options](#) object.

Remarks

The value of the [UseDiffDiacColor](#) property must be **True** in order to use this property.

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the color for diacritics to bright green.

```
If Options.UseDiffDiacColor = True Then _  
    Options.DiacriticColorVal = wdColorBrightGreen
```

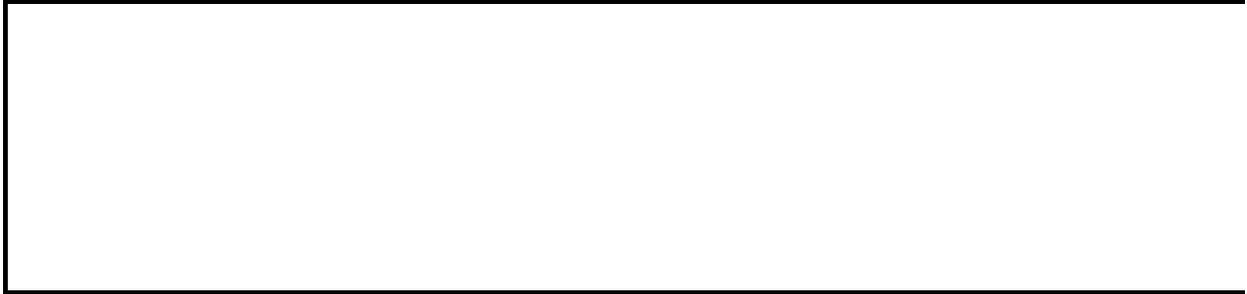


Diagram Property

Returns a [Diagram](#) object to which a diagram node belongs.

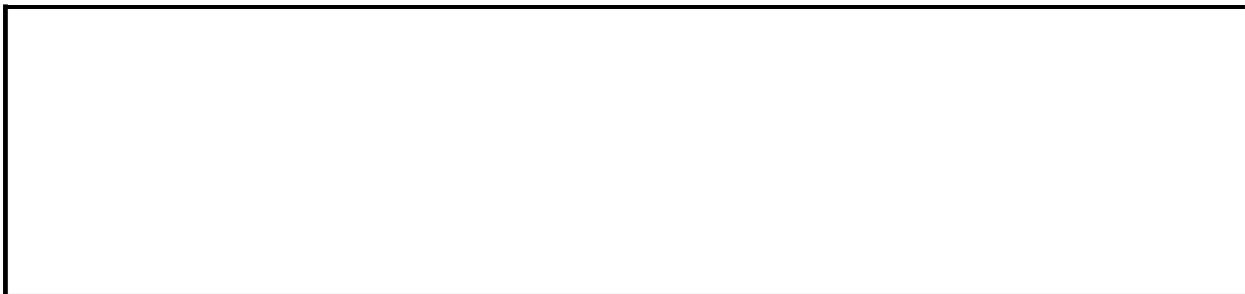
expression.**Diagram**

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

Example

This example converts a pyramid diagram into a radial diagram.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add first child node  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three more child nodes  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    With dgnNode.Diagram  
        'Turn on automatic formatting  
        .AutoFormat = msoTrue  
  
        'Convert pyramid diagram into a radial diagram  
        .Convert Type:=msoDiagramRadial  
    End With  
End Sub
```



DiagramNode Property

Returns a [DiagramNode](#) object that represents a node in a diagram. Read-only.

expression.**DiagramNode**

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

Example

This example adds a pyramid chart to the current document.

```
Sub CreatePyramidDiagram()  
    Dim dgnNode As DiagramNode  
    Dim shpDiagram As Shape  
    Dim intCount As Integer  
  
    'Add pyramid diagram to current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add first diagram node child  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three more diagram child nodes  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
End Sub
```



Dialogs Property

Returns a [Dialogs](#) collection that represents all the built-in dialog boxes in Word. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the built-in **Find** dialog box, with "Hello" in the **Find What** box.

```
Dim dlgFind As Dialog
Set dlgFind = Dialogs(wdDialogEditFind)
With dlgFind
    .Find = "Hello"
    .Show
End With
```

This example displays the built-in **Open** dialog box showing all file types.

```
With Dialogs(wdDialogFileOpen)
    .Name = "*.*)"
    .Show
End With
```

This example prints the active document, using the settings from the **Print** dialog box.

```
Dialogs(wdDialogFilePrint).Execute
```



DifferentFirstPageHeaderFooter Property

True if a different header or footer is used on the first page. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

Example

This example checks each section in the active document for headers and footers that are different on the first page and displays a message if any are found.

```
Dim secLoop As Section
```

```
For Each secLoop In ActiveDocument.Sections
```

```
    If secLoop.PageSetup _
```

```
        .DifferentFirstPageHeaderFooter = True Then
```

```
        MsgBox "Section " & secLoop.Index _
```

```
            & " has different first page headers & footers."
```

```
    End If
```

```
Next secLoop
```



DisableCharacterSpaceGrid Property

-

True if Microsoft Word ignores the number of characters per line for the corresponding **Font** or **Range** object. Returns **wdUndefined** if the **DisableCharacterSpaceGrid** property is set to **True** for only some of the specified font or range. Read/write **Boolean**.

Example

This example signals Microsoft Word to ignore the number of characters per line for the selected text.

```
With Selection.Font  
    .DisableCharacterSpaceGrid = True  
End With
```



DisableFeatures Property

-
True disables all features introduced after the version specified in the [DisableFeaturesIntroducedAfter](#) property. The default value is **False**.
Read/write **Boolean**.

expression.**DisableFeatures**

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

Remarks

The **DisableFeatures** property only affects the document for which you set the property. Use this property if you plan on sharing a document between users with an earlier versions of Microsoft Word, so you don't end up introducing into a document features that are not available in their versions of Word.

Use the [**DisableFeaturesByDefault**](#) property to disable in all documents features introduced after a specified version.

Example

This example disables all features added after Word for Windows 95, versions 7.0 and 7.0a, for the current document. The global default setting remains unchanged.

```
Sub FeaturesDisable()  
  With ThisDocument  
  
    'Checks whether features are disabled  
    If .DisableFeatures = True Then  
  
      'If they are, disables all features after Word for Windo  
      .DisableFeaturesIntroducedAfter = wd70  
    Else  
  
      'If not, turns on the disable features option and disabl  
      'all features introduced after Word for Windows 95  
      .DisableFeatures = True  
      .DisableFeaturesIntroducedAfter = wd70  
    End If  
  End With  
End Sub
```



DisableFeaturesbyDefault Property

-
True for Microsoft Word to disable in all documents all features introduced after the version of Word specified in the

[DisableFeaturesIntroducedAfterByDefault](#). The default value is **False**.

Read/write **Boolean**.

expression.**DisableFeaturesbyDefault**

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

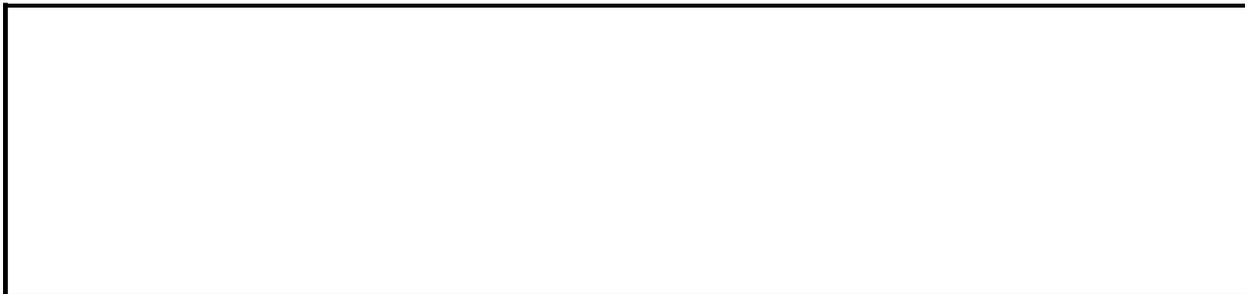
Remarks

The **DisableFeaturesByDefault** property sets a global option for the application. If you want to disable features introduced after Word 97 for Windows for the document only, use the [DisableFeatures](#) property.

Example

This example disables all features introduced after Word for Windows 95, versions 7.0 and 7.0a, for all documents.

```
Sub FeaturesDisableByDefault()  
  With Application.Options  
  
    'Checks whether features are disabled  
    If .DisableFeaturesbyDefault = True Then  
  
      'If they are, disables all features after Word for Windo  
      .DisableFeaturesIntroducedAfterbyDefault = wd70  
    Else  
  
      'If not, turns on the disable features option and disabl  
      'all features introduced after Word for Windows 95  
      .DisableFeaturesbyDefault = True  
      .DisableFeaturesIntroducedAfterbyDefault = wd70  
    End If  
  End With  
End Sub
```



↳ [Show All](#)

DisableFeaturesIntroducedAfter Property

-

Disables all features introduced after a specified version of Microsoft Word in the document only. Read/write [WdDisableFeaturesIntroducedAfter](#).

WdDisableFeaturesIntroducedAfter can be one of these
WdDisableFeaturesIntroducedAfter constants.

wd70 Specifies Word for Windows 95, versions 7.0 and 7.0a.

wd70FE Specifies Word for Windows 95, versions 7.0 and 7.0a, Asian edition.

wd80 Specifies Word 97 for Windows. Default.

expression.**DisableFeaturesIntroducedAfter**

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

Remarks

The [DisableFeatures](#) property must be set to **True** prior to setting the **DisableFeaturesIntroducedAfter** property. Otherwise, the setting will not take effect and will remain at its default setting of Word 97 for Windows.

The **DisableFeaturesIntroducedAfter** property only affects the document for which the property is set. If you want to set a global option for the application to disable features for all documents, use the [DisableFeaturesIntroducedAfterByDefault](#) property.

Example

This example disables all features added after Word for Windows 95, versions 7.0 and 7.0a, for the current document only. The global default setting remains unchanged.

```
Sub FeaturesDisable()  
  With ThisDocument  
  
    'Checks whether features are disabled  
    If .DisableFeatures = True Then  
  
      'If they are, disables all features after Word for Windo  
      .DisableFeaturesIntroducedAfter = wd70  
    Else  
  
      'If not, turns on the disable features option and disabl  
      'all features introduced after Word for Windows 95  
      .DisableFeatures = True  
      .DisableFeaturesIntroducedAfter = wd70  
    End If  
  End With  
End Sub
```



↳ [Show All](#)

DisableFeaturesIntroducedAfterbyDefault Property

-
Disables all features introduced after a the specified version for all documents.
Read/write [WdDisableFeaturesIntroducedAfter](#).

WdDisableFeaturesIntroducedAfter can be one of these
WdDisableFeaturesIntroducedAfter constants.

wd70 Specifies Word for Windows 95, versions 7.0 and 7.0a.

wd70FE Specifies Word for Windows 95, versions 7.0 and 7.0a, Asian edition.

wd80 Specifies Word 97 for Windows. Default.

expression.**DisableFeaturesIntroducedAfterbyDefault**

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

Remarks

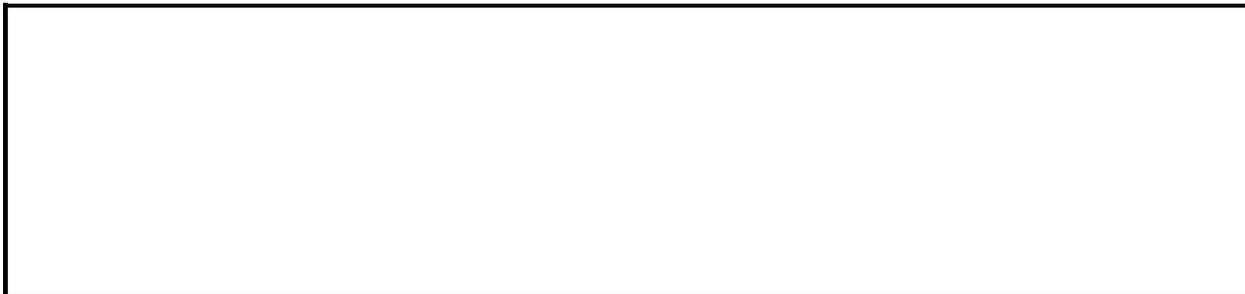
The [DisableFeaturesByDefault](#) property must be set to **True** prior to setting the **DisableFeaturesIntroducedAfterByDefault** property. Otherwise, the setting will not take effect and will remain at its default setting of Word 97 for Windows.

The **DisableFeaturesIntroducedAfterByDefault** property sets a global option for the application and affects all documents. If you want to disable features introduced after a specified version for a document only, use the [DisableFeaturesIntroducedAfter](#) property.

Example

This example disables all features introduced after Word for Windows 95, versions 7.0 and 7.0a, for all documents.

```
Sub FeaturesDisableByDefault()  
  With Application.Options  
  
    'Checks whether features are disabled  
    If .DisableFeaturesbyDefault = True Then  
  
      'If they are, disables all features after Word for Windo  
      .DisableFeaturesIntroducedAfterbyDefault = wd70  
    Else  
  
      'If not, turns on the disable features option and disabl  
      'all features introduced after Word for Windows 95  
      .DisableFeaturesbyDefault = True  
      .DisableFeaturesIntroducedAfterbyDefault = wd70  
    End If  
  End With  
End Sub
```



DisableLineHeightGrid Property

-

True if Microsoft Word aligns characters in the specified paragraphs to the line grid when a set number of lines per page is specified. Returns **wdUndefined** if the **DisableLineHeightGrid** property is set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example sets Microsoft Word to align characters in the selected paragraphs to the line grid if you've specified a set number of lines per page.

```
With Selection.ParagraphFormat  
    .DisableLineHeightGrid = True  
End With
```



DisplayAlerts Property

Returns or sets the way certain alerts and messages are handled while a macro is running. Read/write [WdAlertLevel](#).

WdAlertLevel can be one of these WdAlertLevel constants.

wdAlertsAll All message boxes and alerts are displayed; errors are returned to the macro.

wdAlertsMessageBox Only message boxes are displayed; errors are trapped and returned to the macro.

wdAlertsNone No alerts or message boxes are displayed. If a macro encounters a message box, the default value is chosen and the macro continues.

expression.**DisplayAlerts**

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

Note If you set this property to **wdAlertsNone** or **wdAlertsMessageBox**, Word doesn't set it back to **wdAlertsAll** when your macro stops running. You should write your macro in such a way that it always sets the **DisplayAlerts** property back to **wdAlertsAll** when it stops running.

Example

This example sets Word to display all alerts and message boxes when it's running macros.

```
Application.DisplayAlerts = wdAlertsAll
```

This example returns the current setting of the **DisplayAlerts** property.

```
Dim lngTemp As Long
```

```
lngTemp = Application.DisplayAlerts
```



DisplayAsIcon Property

-

True if the specified object is displayed as an icon. Read/write **Boolean**.

Example

This example displays a message box containing the name of each floating shape that's displayed as an icon on the active document.

```
Dim shapeLoop As Shape

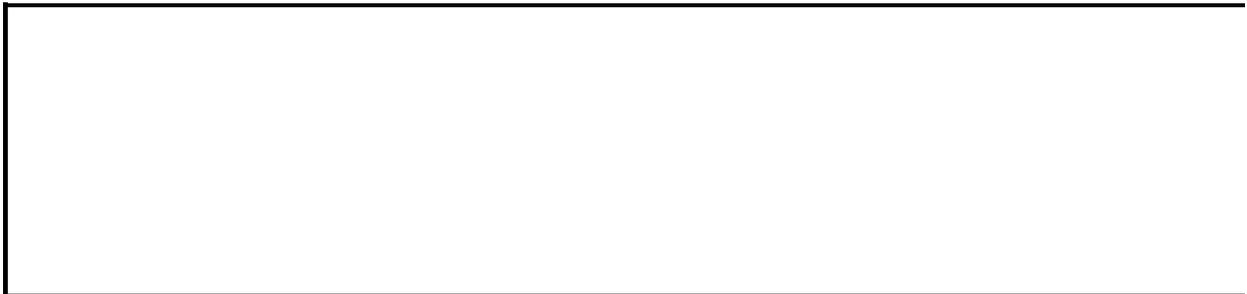
For Each shapeLoop In ActiveDocument.Shapes
    If shapeLoop.OLEFormat.DisplayAsIcon Then
        MsgBox shapeLoop.Name & " is displayed as an icon."
    End If
Next shapeLoop
```

This example inserts a Microsoft Excel worksheet as a linked OLE object on the active document and then changes the display of the object to an icon.

```
Dim objNew As Object

Set objNew = ActiveDocument.Shapes.AddOLEObject _
    (FileName:="C:\Program Files\Microsoft Office" _
    & "\Office\Samples\samples.xls", LinkToFile:=True)

objNew.OLEFormat.DisplayAsIcon = True
```



DisplayAutoCompleteTips Property

-

True if Word displays tips that suggest text for completing words, dates, or phrases as you type. Read/write **Boolean**.

Example

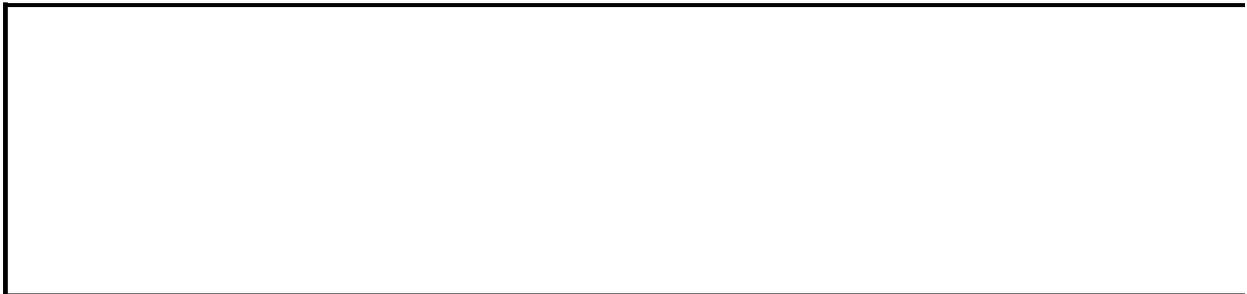
This example sets Word to display tips that suggest text for completing words, dates, or phrases as you type.

```
Application.DisplayAutoCompleteTips = True
```

This example returns the status of the **Suggest the rest of the word or date with a tip as you type** option on the **AutoText** tab in the **AutoCorrect** dialog box (**Tools** menu).

```
Dim blnTemp As Boolean
```

```
blnTemp = Application.DisplayAutoCompleteTips
```



DisplayAutoCorrectOptions Property

True for Microsoft Word to display the **AutoCorrect Options** button.
Read/write **Boolean**.

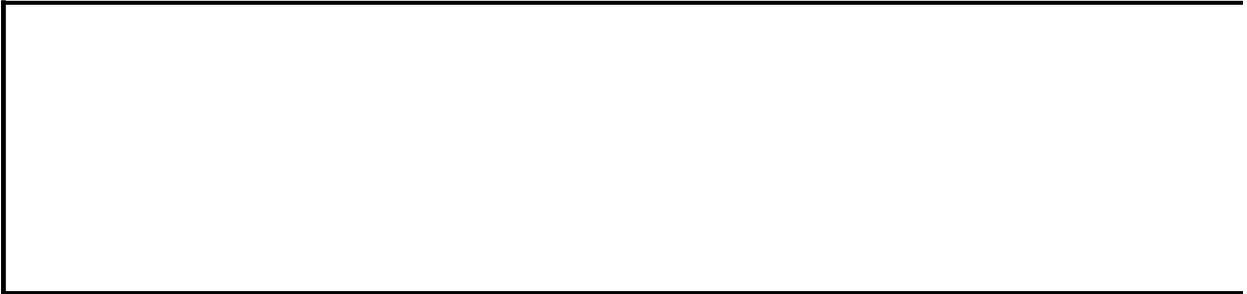
expression.DisplayAutoCorrectOptions

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example disables display of the **AutoCorrect Options** button.

```
Sub HideAutoCorrectOpButton()  
    AutoCorrect.DisplayAutoCorrectOptions = False  
End Sub
```



DisplayGridLines Property

-

True if Microsoft Word displays the document grid. This property is the equivalent of the **Gridlines** command on the **View** menu. Read/write **Boolean**.

Remarks

This property affects only the document grid. For table gridlines, use the [TableGridlines](#) property.

Example

This example switches between displaying and hiding the document grid in the active window.

```
Options.DisplayGridLines = Not Options.DisplayGridLines
```



DisplayHorizontalScrollBar Property

-
True if a horizontal scroll bar is displayed for the specified window. Read/write **Boolean**.

Example

This example displays vertical and horizontal scroll bars for the active window.

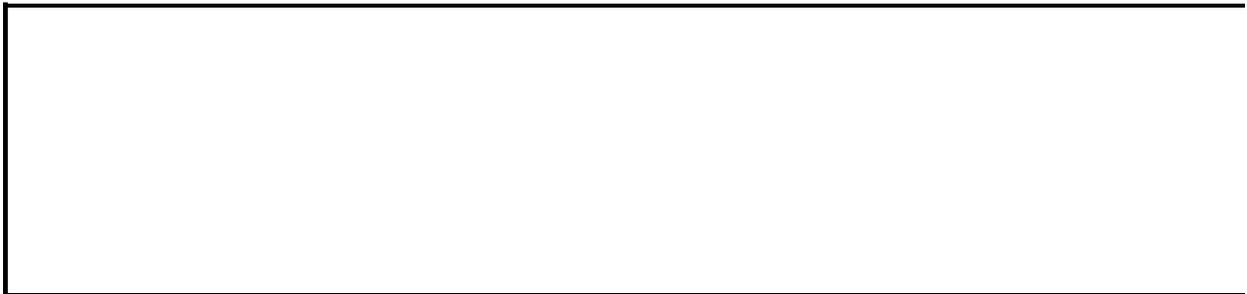
```
With ActiveDocument.ActiveWindow
    .DisplayHorizontalScrollBar = True
    .DisplayVerticalScrollBar = True
End With
```

This example toggles the horizontal scroll bar of the window for Document1.

```
Dim winTemp As Window

Set winTemp = Windows("Document1")

winTemp.DisplayHorizontalScrollBar = _
    Not winTemp.DisplayHorizontalScrollBar
```



DisplayLeftScrollBar Property

True if the vertical scroll bar appears on the left side of the document window.
Read/write **Boolean**.

expression.**DisplayLeftScrollBar**

expression Required. An expression that returns a [Window](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example displays the vertical scroll bar on the left side of the active window.

```
ActiveWindow.DisplayLeftScrollBar = True
```



DisplayPageBoundaries Property

True to display the top and bottom margins (white space) and the gray area (gray space) between pages in a document. **False** to hide from view the white and gray space so that the pages flow together as one long page. The default value is **True**. Read/write **Boolean**.

expression.**DisplayPageBoundaries**

expression Required. An expression that returns a [View](#) object.

Remarks

This feature is only available in the Print Layout view and only affects the gray space on the top and bottom of a page, not the left and right sides of a page. This setting affects the document in the in the specified window. When the document is saved, the state of this setting is saved with it.

Example

This example changes the current view to Print Layout and suppresses the white and gray space between document pages.

```
Sub WhiteSpace()  
  With ActiveWindow.View  
    .Type = wdPrintView  
    .DisplayPageBoundaries = False  
  End With  
End Sub
```



DisplayPasteOptions Property

True for Microsoft Word to display the **Paste Options** button, which displays directly under newly pasted text. Read/write **Boolean**.

expression.**DisplayPasteOptions**

expression Required. An expression that returns an [Options](#) object.

Example

This example enables the **Paste Options** button if the option has been disabled.

```
Sub ShowPasteOptionsButton()  
  With Options  
    If .DisplayPasteOptions = False Then  
      .DisplayPasteOptions = True  
    End If  
  End With  
End Sub
```



DisplayRecentFiles Property

-
True if the names of recently used files are displayed on the **File** menu.
Read/write **Boolean**.

Example

This example sets Word to display a maximum of six file names on the **File** menu.

```
Application.DisplayRecentFiles = True  
RecentFiles.Maximum = 6
```

This example removes the list of recently used files from the **File** menu.

```
Application.DisplayRecentFiles = False
```



DisplayRightRuler Property

True if the vertical ruler appears on the right side of the document window in print layout view. Read/write **Boolean**.

expression.**DisplayRightRuler**

expression Required. An expression that returns a [Window](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the active window to print layout view and displays the vertical ruler on the right side.

```
With ActiveWindow  
    .View = wdPrintView  
    .DisplayRightRuler = True  
End With
```



DisplayRulers Property

-
True if rulers are displayed for the specified window or pane. Equivalent to the **Ruler** command on the **View** menu. Read/write **Boolean**.

Note If **DisplayRulers** is **False**, the horizontal and vertical rulers won't be displayed, regardless of the state of the **DisplayVerticalRuler** property.

Example

This example toggles the ruler display for the active window.

```
ActiveDocument.ActiveWindow.DisplayRulers = _  
    Not ActiveDocument.ActiveWindow.DisplayRulers
```

This example switches the window to print layout view and displays the horizontal and vertical rulers.

```
With ActiveDocument.ActiveWindow  
    .View.Type = wdPrintView  
    .DisplayVerticalRuler = True  
    .DisplayRulers = True  
End With
```



DisplayScreenTips Property

-

True if comments, footnotes, endnotes, and hyperlinks are displayed as tips. Text marked as having comments is highlighted. Read/write **Boolean**.

Example

This example enables Word to display comments, footnotes, and endnotes as tips. Also, text marked as having comments is highlighted.

```
Application.DisplayScreenTips = True
```

This example returns the current status of the **ScreenTips** checkbox in the **Show** area on the **View** tab in the **Options** dialog box.

```
temp = Application.DisplayScreenTips
```



DisplayScrollBars Property

-

True if Word displays a scroll bar in at least one document window. **False** if there are no scroll bars displayed in any window. Read/write **Boolean**.

Remarks

Setting the **DisplayScrollBars** property to **True** displays horizontal and vertical scroll bars in all windows. Setting this property to **False** turns off all scroll bars in all windows.

Use the **DisplayHorizontalScrollBar** and **DisplayVerticalScrollBar** properties to display individual scroll bars in the specified window.

Example

This example displays horizontal and vertical scroll bars in all windows.

```
Application.DisplayScrollBars = True
```

This example returns **True** if there's a scroll bar currently displayed in any window.

```
Dim blnTemp As Boolean
```

```
blnTemp = Application.DisplayScrollBars
```



DisplaySmartTagButtons Property

True for Microsoft Word to display a button directly above a smart tag when a mouse pointer is positioned over it. Read/write **Boolean**.

expression.**DisplaySmartTagButtons**

expression Required. An expression that returns an [Options](#) object.

Remarks

The smart tag button provides a drop-down menu from which a user can access smart tag options and actions.

Example

This example hides the button that appears when the mouse pointer is positioned over a smart tag.

```
Sub DontShowSmartTagButton()  
    Options.DisplaySmartTagButtons = False  
End Sub
```



DisplaySmartTags Property

True for Microsoft Word to display an underline beneath smart tags in a document. Read/write **Boolean**.

expression.**DisplaySmartTags**

expression Required. An expression that returns a [View](#) object.

Remarks

Smart tags are marked in documents with a dashed underline. Setting the **DisplaySmartTags** property to **False** does not remove smart tags; it only turns off displaying the underline.

Example

This example turns off displaying the underline beneath smart tags in the active view.

```
Sub DontShowSmartTags()  
    ActiveWindow.View.ShowSmartTags = False  
End Sub
```



DisplayStatusBar Property

-

True if the status bar is displayed. Read/write **Boolean**.

Example

This example toggles the status bar.

```
Application.DisplayStatusBar = Not Application.DisplayStatusBar
```

This example displays scroll bars and the status bar.

```
With Application  
    .DisplayScrollBars = True  
    .DisplayStatusBar = True  
End With
```



DisplayVerticalRuler Property

True if a vertical ruler is displayed for the specified window or pane. Read/write **Boolean**.

Note A vertical ruler appears only in print layout view, and only if the **DisplayRulers** property is set to **True**.

Example

This example switches each window in the **Windows** collection to print layout view and displays the horizontal and vertical rulers.

```
Dim winLoop As Window

For Each winLoop In Windows
    With winLoop
        .View.Type = wdPrintView
        .DisplayRulers = True
        .DisplayVerticalRuler = True
    End With
Next winLoop
```

This example hides the horizontal and vertical rulers for the active window.

```
With ActiveDocument.ActiveWindow
    .DisplayVerticalRuler = False
    .DisplayRulers = False
End With
```



DisplayVerticalScrollBar Property

-
True if a vertical scroll bar is displayed for the specified window. Read/write **Boolean**.

Example

This example displays the vertical and horizontal scroll bars for each window in the **Windows** collection.

```
Dim winLoop As Window

For Each winLoop In Windows
    winLoop.DisplayVerticalScrollBar = True
    winLoop.DisplayHorizontalScrollBar = True
Next winLoop
```

This example toggles the vertical scroll bar for the active window.

```
Dim winTemp As Window

Set winTemp = ActiveDocument.ActiveWindow
winTemp.DisplayVerticalScrollBar = _
    Not winTemp.DisplayVerticalScrollBar
```



DistanceBottom Property

-

Rows object: Returns or sets the distance (in points) between the document text and the bottom edge of the specified table. This property doesn't have any effect if [WrapAroundText](#) is False. Read/write **Single**.

WrapFormat object: Returns or sets the distance (in points) between the document text and the bottom edge of the text-free area surrounding the specified shape. The size and shape of the specified shape, together with the values of the **Type** and **Side** properties of the **WrapFormat** object, determine the size and shape of this text-free area. Read/write **Single**.

Example

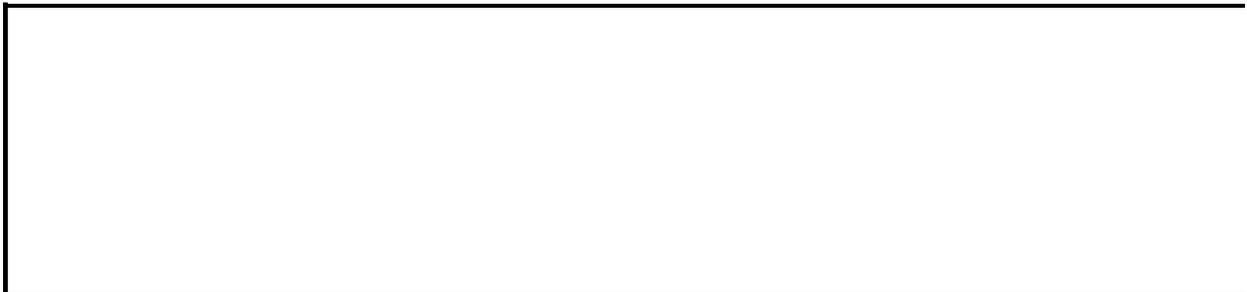
This example sets text to wrap around the first table in the active document and sets the distance for wrapped text to 20 points on all sides of the table.

```
With ActiveDocument.Tables(1).Rows
    .WrapAroundText = True
    .DistanceLeft = 20
    .DistanceRight = 20
    .DistanceTop = 20
    .DistanceBottom = 20
End With
```

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.

```
Dim shapeOval As Shape

Set shapeOval = ActiveDocument.Shapes.AddShape(msoShapeOval, _
    36, 36, 100, 35)
With shapeOval.WrapFormat
    .Type = wdWrapSquare
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



DistanceFrom Property

Returns or sets a value that indicates whether the specified page border is measured from the edge of the page or from the text it surrounds. Read/write [WdBorderDistanceFrom](#).

WdBorderDistanceFrom can be one of these WdBorderDistanceFrom constants.

wdBorderDistanceFromPageEdge

wdBorderDistanceFromText

expression.**DistanceFrom**

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

Example

This example adds a single border around each page in section one in the active document and then sets the distance between each border and the corresponding edge of the page.

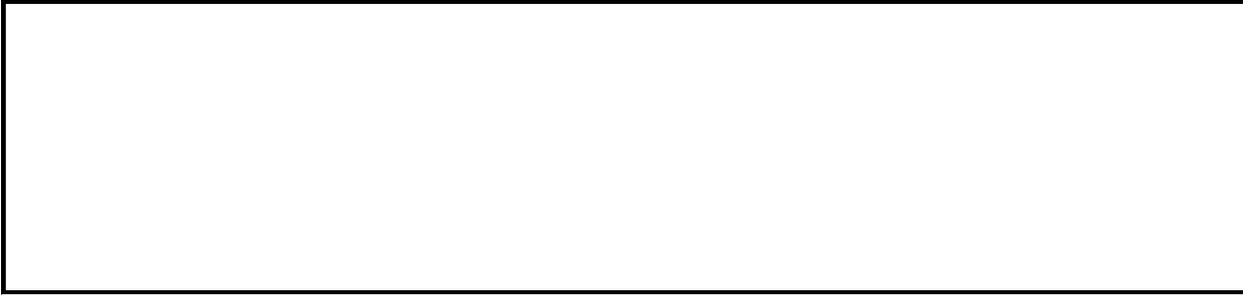
```
Dim borderLoop As Border

With ActiveDocument.Sections(1)
    For Each borderLoop In .Borders
        borderLoop.LineStyle = wdLineStyleSingle
        borderLoop.LineWidth = wdLineWidth050pt
    Next borderLoop
    With .Borders
        .DistanceFrom = wdBorderDistanceFromPageEdge
        .DistanceFromTop = 20
        .DistanceFromLeft = 22
        .DistanceFromBottom = 20
        .DistanceFromRight = 22
    End With
End With
```

This example adds a border around each page in the first section in the selection, and then it sets the distance between the text and the page border to 6 points.

```
Dim borderLoop As Border

With Selection.Sections(1)
    For Each borderLoop In .Borders
        borderLoop.ArtStyle = wdArtSeattle
        borderLoop.ArtWidth = 22
    Next borderLoop
    With .Borders
        .DistanceFrom = wdBorderDistanceFromText
        .DistanceFromTop = 6
        .DistanceFromLeft = 6
        .DistanceFromBottom = 6
        .DistanceFromRight = 6
    End With
End With
```



DistanceFromBottom Property

Returns or sets the space (in points) between the text and the bottom border.
Read/write **Long**.

Note Using this property with a page border, you can set either the space between the text and the bottom page border or the space between the bottom edge of the page and the bottom page border. Where the distance is measured from depends on the value of the [DistanceFrom](#) property.

Example

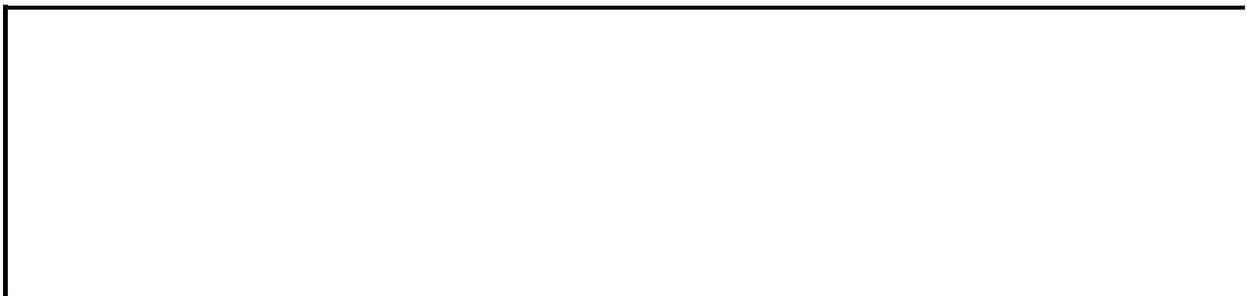
This example adds a border around the first paragraph in the active document and sets the distance between the text and the bottom border to 6 points.

```
With ActiveDocument.Paragraphs(1).Borders
    .Enable = True
    .DistanceFromBottom = 6
End With
```

This example adds a border around each table in Sales.doc. The example also sets the distance between the text and the border to 3 points for the top and bottom borders, and 6 points for the left and right borders.

```
Dim tableLoop As Table

For Each tableLoop In Documents("Sales.doc").Tables
    With tableLoop.Borders
        .OutsideLineStyle = wdLineStyleSingle
        .OutsideLineWidth = wdLineWidth150pt
        .DistanceFromBottom = 3
        .DistanceFromTop = 3
        .DistanceFromLeft = 6
        .DistanceFromRight = 6
    End With
Next tableLoop
```



DistanceFromLeft Property

Returns or sets the space (in points) between the text and the left border.
Read/write **Long**.

Note Using this property with a page border, you can set either the space between the text and the left page border or the space between the left edge of the page and the left page border. Where the distance is measured from depends on the value of the [DistanceFrom](#) property.

Example

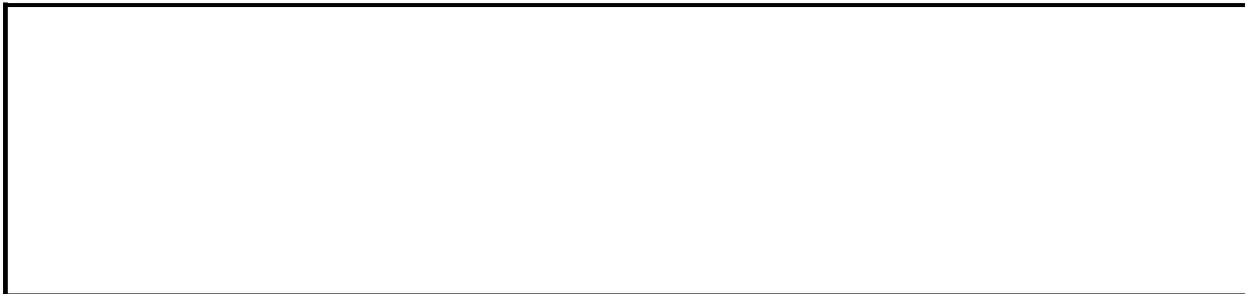
This example adds a border around each frame in the active document and sets the distance between the frame and the border to 5 points.

```
Dim frameLoop As Frame

For Each frameLoop In ActiveDocument.Frames
    With frameLoop.Borders
        .Enable = True
        .DistanceFromLeft = 5
        .DistanceFromRight = 5
        .DistanceFromTop = 5
        .DistanceFromBottom = 5
    End With
Next frameLoop
```

This example adds a border around the first paragraph in the active document and sets the distance between the text and the left border to 3 points.

```
With ActiveDocument.Paragraphs(1).Borders
    .Enable = True
    .DistanceFromLeft = 3
End With
```



DistanceFromRight Property

Returns or sets the space (in points) between the right edge of the text and the right border. Read/write **Long**.

Note Using this property with a page border, you can set either the space between the text and the right border or the space between the right edge of the page and the right border. Where the distance is measured from depends on the value of the [DistanceFrom](#) property.

Example

This example adds a border around each paragraph in the selection and sets the distance between the text and the right border to 3 points.

```
With Selection.Paragraphs.Borders
    .Enable = True
    .DistanceFromRight = 3
End With
```

This example adds a single border around each page in section one in the active document. The example also sets the distance between the right and left border and the corresponding edges of the page to 22 points.

```
Dim borderLoop As Border

With ActiveDocument.Sections(1)
    For Each borderLoop In .Borders
        borderLoop.LineStyle = wdLineStyleSingle
        borderLoop.LineWidth = wdLineWidth050pt
    Next borderLoop
    With .Borders
        .DistanceFrom = wdBorderDistanceFromPageEdge
        .DistanceFromLeft = 22
        .DistanceFromRight = 22
    End With
End With
```



DistanceFromText Property

-
DropCap object: Returns or sets the distance (in points) between the dropped capital letter and the paragraph text. Read/write **Single**.

LineNumbering object: Returns or sets the distance (in points) between the right edge of line numbers and the left edge of the document text. Read/write **Single**.

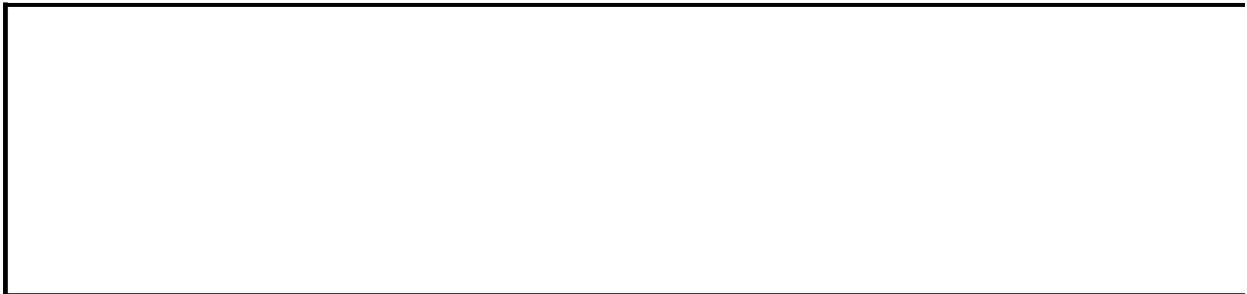
Example

This example adds line numbers to the active document. The distance between the line numbers and the left margin is 36 points (0.5 inch).

```
With ActiveDocument.PageSetup.LineNumbering
    .Active = True
    .CountBy = 5
    .DistanceFromText = 36
End With
```

This example sets a dropped capital letter for the first paragraph in the active document. The offset for the dropped capital letter is then set to 12 points.

```
With ActiveDocument.Paragraphs(1).DropCap
    .Enable
    .FontName= "Arial"
    .Position = wdDropNormal
    .DistanceFromText = 12
End With
```



DistanceFromTop Property

Returns or sets the space (in points) between the text and the top border.
Read/write **Long**.

Note Using this property with a page border, you can set either the space between the text and the top page border or the space between the top edge of the page and the top page border. Where the distance is measured from depends on the value of the [DistanceFrom](#) property.

Example

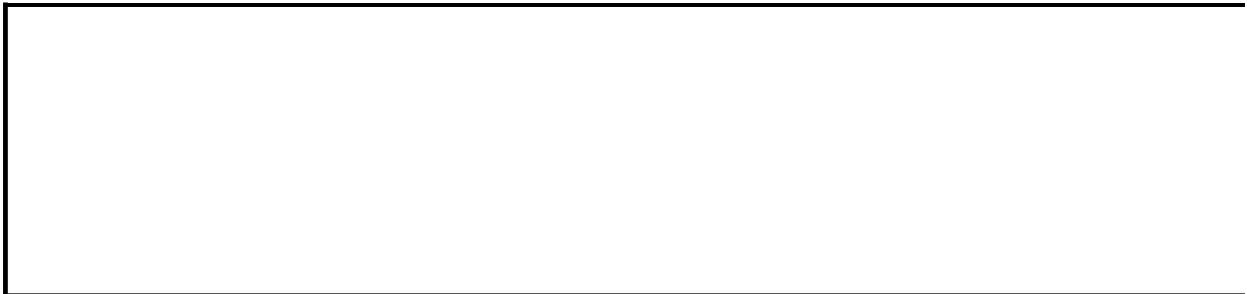
This example adds a border around each paragraph in the selection and sets the distance between the text and the top border to 3 points.

```
With Selection.Borders
    .Enable = True
    .DistanceFromTop = 3
End With
```

This example adds a border around each page in the first section in the selection. The example also sets the distance between the text and the page border to 6 points.

```
Dim borderLoop As Border

With Selection.Sections(1)
    For Each borderLoop In .Borders
        borderLoop.ArtStyle = wdArtSeattle
        borderLoop.ArtWidth = 22
    Next borderLoop
    With .Borders
        .DistanceFrom = wdBorderDistanceFromText
        .DistanceFromTop = 6
        .DistanceFromLeft = 6
        .DistanceFromBottom = 6
        .DistanceFromRight = 6
    End With
End With
```



DistanceLeft Property

-

Rows object: Returns or sets the distance (in points) between the document text and the left edge of the specified table. This property doesn't have any effect if [WrapAroundText](#) is False. Read/write **Single**.

WrapFormat object: Returns or sets the distance (in points) between the document text and the left edge of the text-free area surrounding the specified shape. The size and shape of the specified shape, together with the values of the **Type** and **Side** properties of the **WrapFormat** object, determine the size and shape of this text-free area. Read/write **Single**.

Example

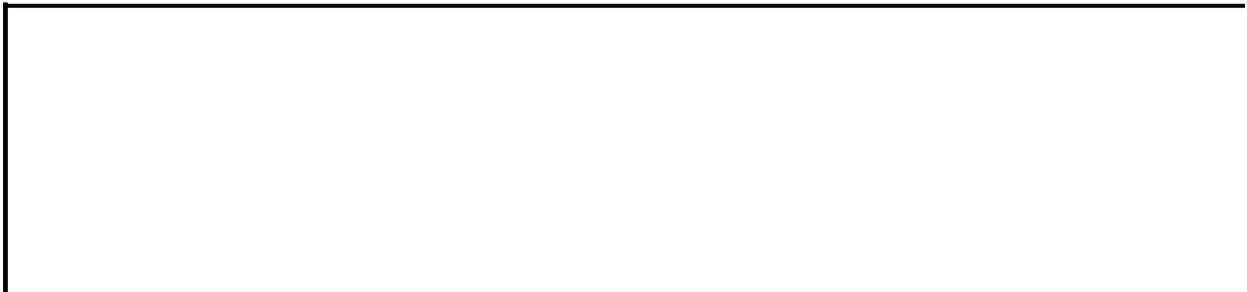
This example sets text to wrap around the first table in the active document and sets the distance for wrapped text to 20 points on all sides of the table.

```
With ActiveDocument.Tables(1).Rows
    .WrapAroundText = True
    .DistanceLeft = 20
    .DistanceRight = 20
    .DistanceTop = 20
    .DistanceBottom = 20
End With
```

This example adds an oval to the active document and specifies that the document text wrap tightly around 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 oval.

```
Dim shapeOval As Shape

Set shapeOval = ActiveDocument.Shapes.AddShape(msoShapeOval, _
    0, 0, 200, 50)
With shapeOval.WrapFormat
    .Type = wdWrapTight
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



DistanceRight Property

-

Rows object: Returns or sets the distance (in points) between the document text and the right edge of the specified table. This property doesn't have any effect if [WrapAroundText](#) is False. Read/write **Single**.

WrapFormat object: Returns or sets the distance (in points) between the document text and the right edge of the text-free area surrounding the specified shape. The size and shape of the specified shape, together with the values of the **Type** and **Side** properties of the **WrapFormat** object, determine the size and shape of this text-free area. Read/write **Single**.

Example

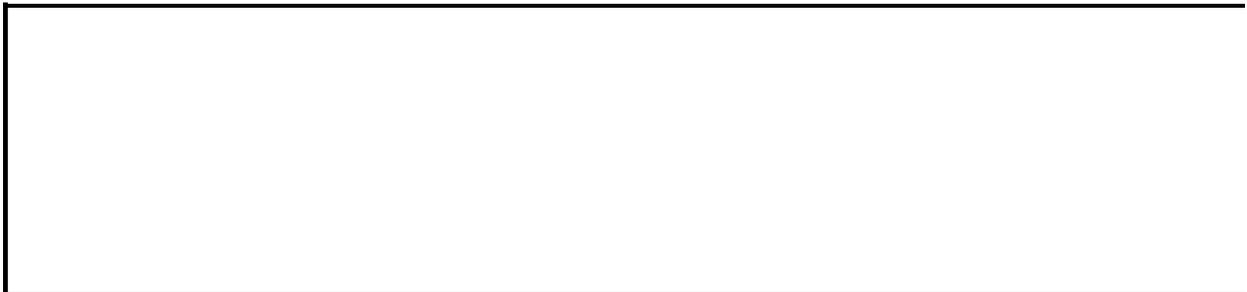
This example sets text to wrap around the first table in the active document and sets the distance for wrapped text to 20 points on all sides of the table.

```
With ActiveDocument.Tables(1).Rows
    .WrapAroundText = True
    .DistanceLeft = 20
    .DistanceRight = 20
    .DistanceTop = 20
    .DistanceBottom = 20
End With
```

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.

```
Dim shapeOval As Shape

Set shapeOval = ActiveDocument.Shapes.AddShape(msoShapeOval, _
    0, 0, 200, 50)
With shapeOval.WrapFormat
    .Type = wdWrapSquare
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



DistanceTop Property

-

Rows object: Returns or sets the distance (in points) between the document text and the top edge of the specified table. This property doesn't have any effect if [WrapAroundText](#) is False. Read/write **Single**.

WrapFormat object: Returns or sets the distance (in points) between the document text and the top edge of the text-free area surrounding the specified shape. The size and shape of the specified shape, together with the values of the **Type** and **Side** properties of the **WrapFormat** object, determine the size and shape of this text-free area. Read/write **Single**.

Example

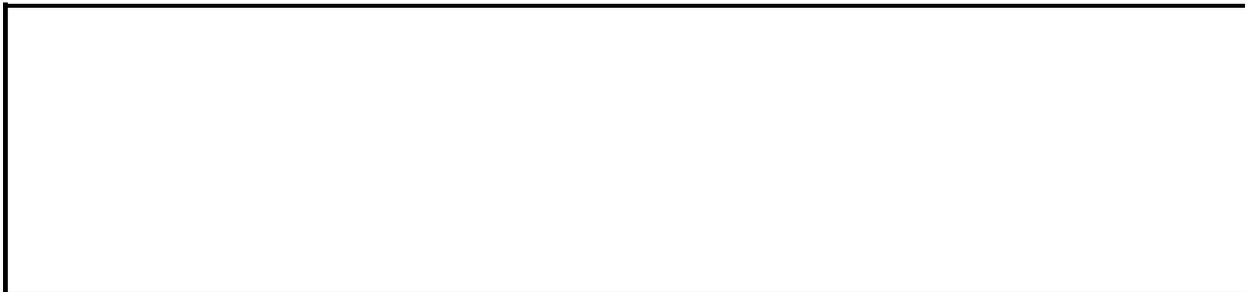
This example sets text to wrap around the first table in the active document and sets the distance for wrapped text to 20 points on all sides of the table.

```
With ActiveDocument.Tables(1).Rows
    .WrapAroundText = True
    .DistanceLeft = 20
    .DistanceRight = 20
    .DistanceTop = 20
    .DistanceBottom = 20
End With
```

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.

```
Dim shapeOval As Shape

Set shapeOval = ActiveDocument.Shapes.AddShape(msoShapeOval, _
    0, 0, 200, 50)
With shapeOval.WrapFormat
    .Type = wdWrapSquare
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



Document Property

-

Returns a [Document](#) object associated with the specified pane, window, or selection. Read-only.

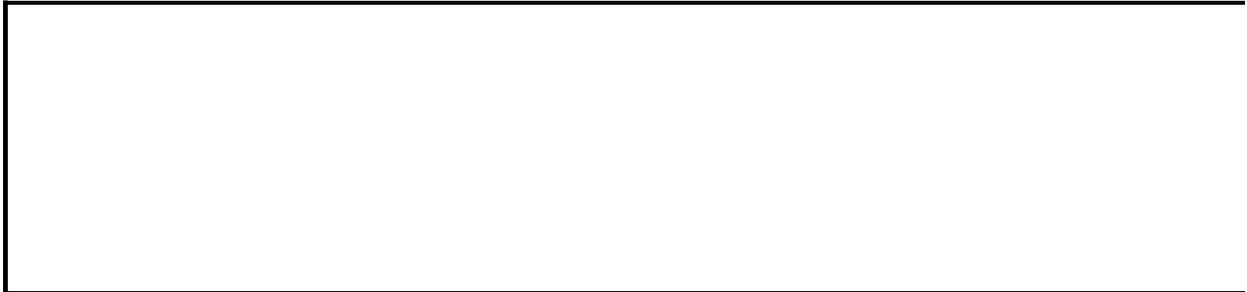
Example

This example displays the document name and path for the selection.

```
Msgbox Selection.Document.FullName
```

This example sets myDoc to the document associated with the active window. The focus is changed to the next window, and the window is split. The **Activate** method is used to switch back to the original document.

```
Set myDoc = Application.ActiveWindow.Document  
If Windows.Count >= 2 Then  
    Application.ActiveWindow.Next.Activate  
    Application.ActiveWindow.Split = True  
    myDoc.Activate  
End If
```



DocumentMap Property

-

True if the document map is visible. Read/write **Boolean**.

Example

This example toggles the document map for the active window.

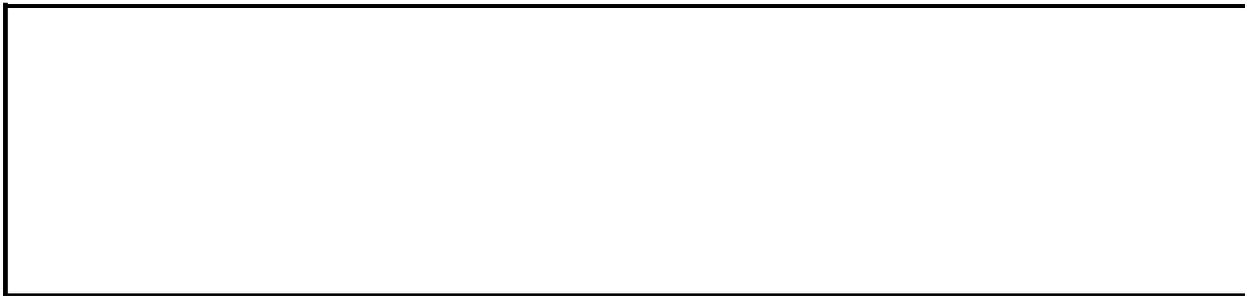
```
ActiveDocument.ActiveWindow.DocumentMap = _  
    Not ActiveDocument.ActiveWindow.DocumentMap
```

This example displays the document map in the window for Sales.doc.

```
Dim docSales As Document
```

```
Set docSales = _  
    Documents.Open(FileName:="C:\Documents\Sales.doc")
```

```
docSales.ActiveWindow.DocumentMap = True
```



DocumentMapPercentWidth Property

Returns or sets the width of the document map as a percentage of the width of the specified window. Read/write **Long**.

Example

This example displays the document map for the active window and sets the map's width to 25 percent of the window's width.

```
With ActiveDocument.ActiveWindow  
    .DocumentMap = True  
    .DocumentMapPercentWidth = 25  
End With
```



Documents Property

-
Returns a [Documents](#) collection that represents all the open documents. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a new document based on the Normal template and then displays the **Save As** dialog box.

```
Documents.Add.Save
```

This example saves open documents that have changed since they were last saved.

```
Dim docLoop As Document
```

```
For Each docLoop In Documents  
    If docLoop.Saved = False Then docLoop.Save  
Next docLoop
```

This example prints each open document after setting the left and right margins to 0.5 inch.

```
Dim docLoop As Document
```

```
For Each docLoop In Documents  
    With docLoop  
        .PageSetup.LeftMargin = InchesToPoints(0.5)  
        .PageSetup.RightMargin = InchesToPoints(0.5)  
        .PrintOut  
    End With  
Next docLoop
```

This example opens Doc.doc as a read-only document.

```
Documents.Open FileName:="C:\Files\Doc.doc", ReadOnly:=True
```



▾ [Show All](#)

DocumentViewDirection Property

Returns or sets the alignment and reading order for the entire document.
Read/write [WdDocumentViewDirection](#).

WdDocumentViewDirection can be one of these WdDocumentViewDirection constants.

wdDocumentViewLtr Displays the document with left alignment and left-to-right reading order.

wdDocumentViewRtl Displays the document with right alignment and right-to-left reading order.

expression.**DocumentViewDirection**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the alignment to right and the reading order to right-to-left for the entire document.

```
Options.DocumentViewDirection = wdDocumentViewRtl
```



DoNotEmbedSystemFonts Property

True for Microsoft Word to not embed common system fonts. Read/write **Boolean**.

expression.**DoNotEmbedSystemFonts**

expression Required. An expression that returns a [Document](#) object.

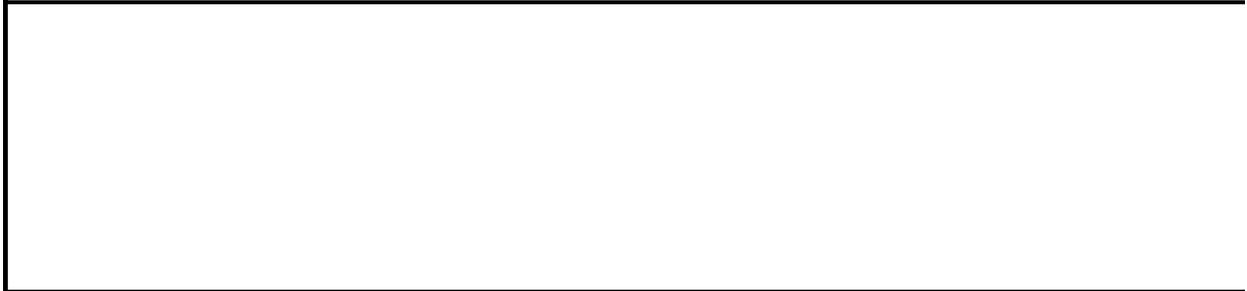
Remarks

Setting the **DoNotEmbedSystemFonts** property to **False** is useful if the user is on an East Asian system and wants to create a document that is readable by others who do not have fonts for that language on their system. For example, a user on a Japanese system could choose to embed the fonts in a document so that the Japanese document would be readable on all systems.

Example

This example embeds all fonts in the current document.

```
Sub EmbedFonts()  
  With ThisDocument  
    If .EmbedTrueTypeFonts = False Then  
      .EmbedTrueTypeFonts = True  
      .DoNotEmbedSystemFonts = False  
    Else  
      .DoNotEmbedSystemFonts = False  
    End If  
  End With  
End Sub
```



DotMatrix Property

-

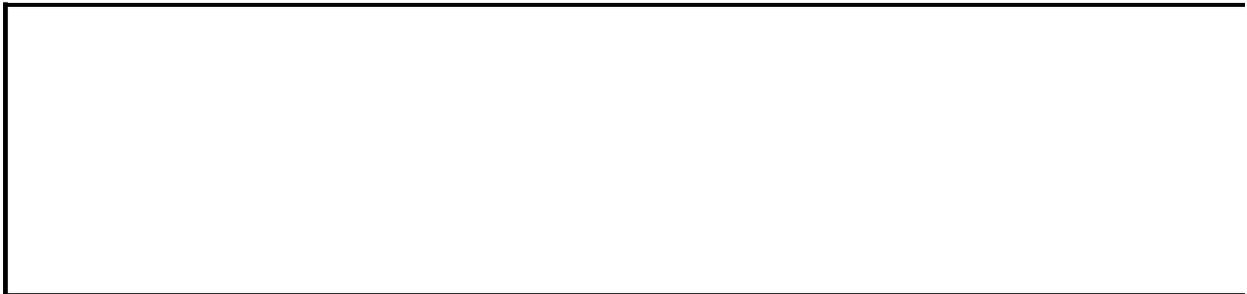
True if the printer type for the specified custom label is dot matrix. **False** if the printer type is either laser or ink jet. Read-only **Boolean**.

Example

This example displays the name and printer type of the first custom mailing label.

```
Dim mlTemp As MailingLabel

Set mlTemp = Application.MailingLabel
If mlTemp.CustomLabels.Count >= 1 Then
    If mlTemp.CustomLabels(1).DotMatrix = True Then
        MsgBox mlTemp.CustomLabels(1).Name & " is dot matrix"
    Else
        MsgBox mlTemp.CustomLabels(1).Name _
            & " is laser or ink jet"
    End If
End If
```



DoubleQuote Property

True if Microsoft Word encloses the specified [PageNumbers](#) object in double quotation marks ("). Read/write **Boolean**.

expression.**DoubleQuote**

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

Remarks

To set Word to enclose page numbers in double quotation marks by default, use the [AddHebDoubleQuote](#) property.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example encloses the page numbers in the first footer of the active document in double quotation marks ("").

```
ActiveDocument.Sections(1).Footers(1) _  
    .PageNumbers.DoubleQuote = True
```



DoubleStrikeThrough Property

True if the specified font is formatted as double strikethrough text. Returns **True**, **False**, or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

Note To set or return single-line strikethrough formatting, use the [StrikeThrough](#) property. Setting **DoubleStrikeThrough** to **True** sets **StrikeThrough** to **False**, and vice versa.

Example

This example applies double strikethrough formatting to the selected text.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.DoubleStrikeThrough = True
Else
    MsgBox "You need to select some text."
End If
```

This example removes double strikethrough formatting from the first word in the active document and capitalizes the first letter in the word.

```
With ActiveDocument.Words(1)
    .Font.DoubleStrikeThrough = False
    .Case = wdTitleSentence
End With
```



DownloadURL Property

Returns a **String** that represents the URL address for a smart tag. Read-only.

expression.**DownloadURL**

expression Required. An expression that returns a [SmartTag](#) object.

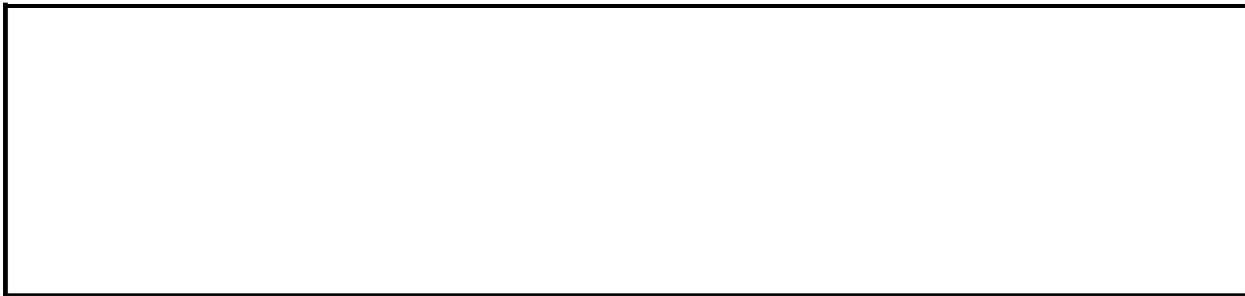
Remarks

The URL address is specified in the related smart tag recognizer file. When a piece of text is recognized and marked, the URL becomes part of the information contained in the smart tag. The **DownloadURL** property is useful if a document is sent to someone who does not have the necessary recognizer and action files installed on their computer. The user can follow the URL to download the necessary smart tag files.

Example

This example loops through the smart tags in the current document and, if a smart tag has a URL address, lists the address in a new document.

```
Sub SmartTagDownloadURL()  
    Dim docNew As Document  
    Dim stgTag As SmartTag  
    Dim intCount As Integer  
  
    Set docNew = Documents.Add  
  
    docNew.Content.InsertAfter "Smart Tag URLs"  
    docNew.Content.InsertParagraphAfter  
  
    For Each stgTag In ThisDocument.SmartTags  
        intCount = intCount + 1  
        If ThisDocument.SmartTags(intCount).DownloadURL <> "" Then  
            docNew.Content.InsertAfter ThisDocument _  
                .SmartTags(intCount).DownloadURL  
            docNew.Content.InsertParagraphAfter  
        End If  
    Next  
  
End Sub
```



Draft Property

-

True if all the text in a window is displayed in the same sans-serif font with minimal formatting to speed up display. Read/write **Boolean**.

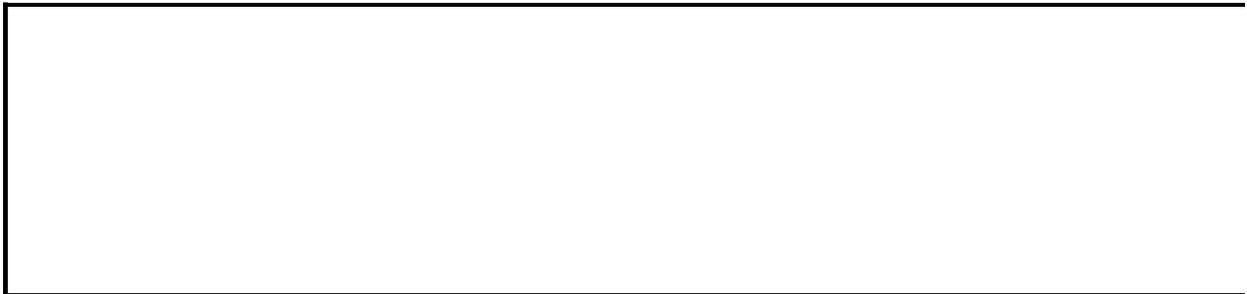
Example

This example displays the contents of the window for Document1 in the draft font.

```
Windows("Document1").View.Draft = True
```

This example toggles the draft font option for the active window.

```
ActiveDocument.ActiveWindow.View.Draft = _  
    Not ActiveDocument.ActiveWindow.View.Draft
```



Drop Property

-

For callouts with an explicitly set drop value, this property returns the vertical distance (in points) 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 (the place that the callout points to), in which case the drop distance is measured from the bottom of the text box. Read-only **Single**.

Remarks

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**. Use the statement `PresetDrop msoCalloutCustomDrop` to set the [DropType](#) property to **msoCalloutDropCustom**.

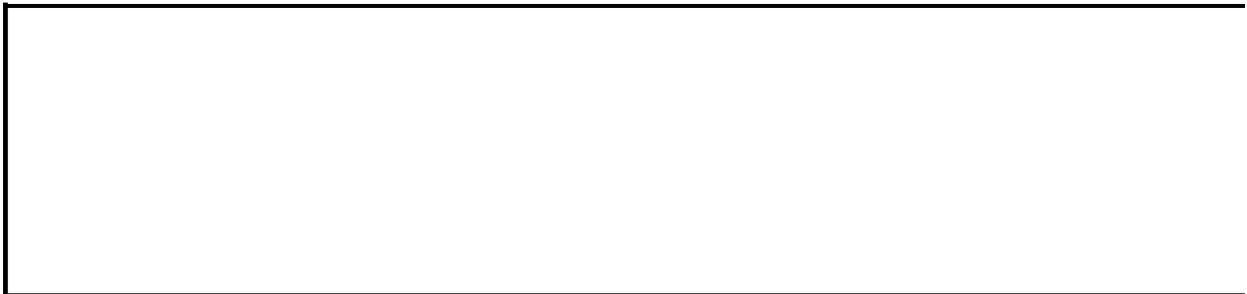
Example

This example replaces the custom drop for the first shape on the active document 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 first shape must be a callout.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes(1).Callout
    If .DropType = msoCalloutDropCustom Then
        If .Drop < .Parent.Height / 2 Then
            .PresetDrop msoCalloutDropTop
        Else
            .PresetDrop msoCalloutDropBottom
        End If
    End If
End With
```



DropCap Property

-

Returns a **DropCap** object that represents a dropped capital letter for the specified paragraph. Read-only.

Example

This example sets a dropped capital letter for the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).DropCap
    .FontName = "Arial"
    .Position = wdDropNormal
    .LinesToDrop = 3
    .DistanceFromText = InchesToPoints(0.1)
End With
```



DropDown Property

-

Returns a [DropDown](#) object that represents a drop-down form field. Read-only.

Remarks

If the **DropDown** property is applied to a **FormField** object that isn't a drop-down form field, the property won't fail, but the [Valid](#) property for the returned object will be **False**.

Example

This example displays the text of the item selected in the drop-down form field named "Colors."

```
Dim ffDrop As FormField
```

```
Set ffDrop = ActiveDocument.FormFields("Colors").DropDown
```

```
MsgBox ffDrop.ListEntries(ffDrop.Value).Name
```

This example adds "Seattle" to the drop-down form field named "Places" in Form.doc.

```
With Documents("Form.doc").FormFields("Places") _  
    .DropDown.ListEntries  
    .Add Name:="Seattle"  
End With
```



DropType Property

Returns a value that indicates where the callout line attaches to the callout text box. Read-only [MsoCalloutDropType](#).

MsoCalloutDropType can be one of these MsoCalloutDropType constants.

msoCalloutDropCenter

msoCalloutDropMixed

msoCalloutDropBottom

msoCalloutDropCustom

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 (the place that the callout points to) are used to determine where the callout line attaches to the text box.

This property is read-only. Use the [PresetDrop](#) method to set the value of this property.

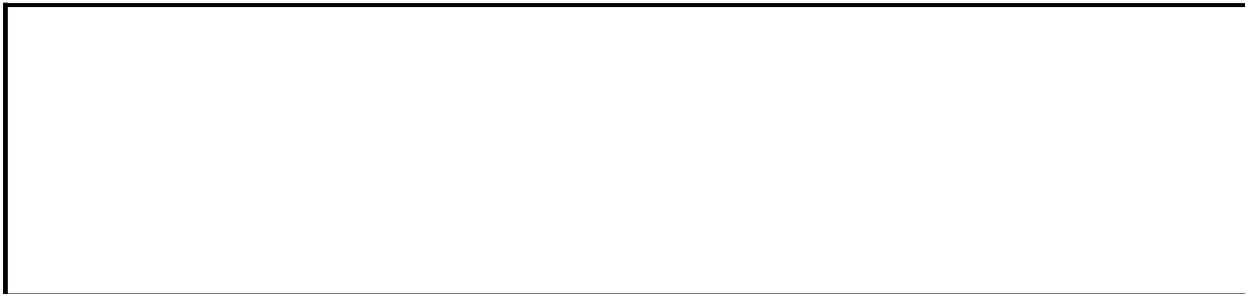
Example

This example checks to determine whether the third shape on the active document is a callout with a custom drop. If it is, the code replaces the custom drop 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.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes(3)
    If .Type = msoCallout Then
        With .Callout
            If .DropType = msoCalloutDropCustom Then
                If .Drop < .Parent.Height / 2 Then
                    .PresetDrop msoCalloutDropTop
                Else
                    .PresetDrop msoCalloutDropBottom
                End If
            End If
        End With
    End If
End With
```



↳ [Show All](#)

Duplicate Property

▶ [Duplicate property as it applies to the **Font** object.](#)

Returns a read-only **Font** object that represents the character formatting of the specified font.

expression.**Duplicate**

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

▶ [Duplicate property as it applies to the **LetterContent** object.](#)

Returns a read-only **LetterContent** object that represents the contents of the specified letter created by the Letter Wizard.

expression.**Duplicate**

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

▶ [Duplicate property as it applies to the **ParagraphFormat** object.](#)

Returns a read-only **ParagraphFormat** object that represents the paragraph formatting of the specified paragraph.

expression.**Duplicate**

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

▶ [Duplicate property as it applies to the **Range** object.](#)

Returns a read-only **Range** object that represents all the properties of the specified range.

expression.**Duplicate**

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

▶ [Duplicate property as it applies to the **TextRetrievalMode** object.](#)

Returns a read-only **TextRetrievalMode** object that represents options related to retrieving text from the specified **Range** object.

expression.**Duplicate**

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

Remarks

You can use the **Duplicate** property to pick up the settings of all the properties of a duplicated **Font**, **LetterContent**, or **ParagraphFormat** object. You can assign the object returned by the **Duplicate** property to another object of the same type to apply those settings all at once. Before assigning the duplicate object to another object, you can change any of the properties of the duplicate object without affecting the original.

By duplicating a **Range** object, you can change the starting or ending character position of the duplicate range without changing the original range.

Example

▶ [As it applies to the **Font** object.](#)

This example sets the variable `MyDupFont` to the character formatting of the selection, removes bold formatting from `MyDupFont`, and adds italic formatting to it instead. The example also creates a new document, inserts text into it, and then applies the formatting stored in `MyDupFont` to the text.

```
Set myDupFont = Selection.Font.Duplicate
With myDupFont
    .Bold = False
    .Italic = True
End With
Documents.Add
Selection.InsertAfter "This is some text."
Selection.Font = myDupFont
```

▶ [As it applies to the **ParagraphFormat** object.](#)

This example duplicates the paragraph formatting of the first paragraph in the active document and stores the formatting in the variable `myDup`, and then it changes the left indent for `myDup` to 1 inch. The example also creates a new document, inserts text into it, and then applies the paragraph formatting stored in `myDup` to the text.

```
ActiveDocument.Range(Start:=0, End:=0).InsertAfter _
    "Paragraph Number 1"
Set myDup = ActiveDocument.Paragraphs(1).Format.Duplicate
myDup.LeftIndent = InchesToPoints(1)
Documents.Add
Selection.InsertAfter "This is a new paragraph."
Selection.Paragraphs.Format = myDup
```



► [As it applies to the **Range** object.](#)

This example duplicates the **Range** object assigned to the variable myRange. The example collapses the duplicate range to its end point, expands it by one character, and makes this character uppercase. The example then applies italic formatting to the original **Range** object (myRange).

```
Set myRange = Selection.Range
With myRange.Duplicate
    .Collapse Direction:=wdCollapseEnd
    .Expand Unit:=wdCharacter
    .Case = wdUpperCase
End With
myRange.Font.Italic = True
```

EditingType Property

-

If the specified node is a vertex, this property returns a value that indicates how changes made to the node affect the two segments connected to the node. Read-only [MsoEditingType](#). If the node is a control point for a curved segment, this property returns the editing type of the adjacent vertex.

MsoEditingType can be one of these MsoEditingType constants.

msoEditingAuto

msoEditingCorner

msoEditingSmooth

msoEditingSymmetric

expression.**EditingType**

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

Remarks

This property is read-only. Use the [SetEditingType](#) method to set the value of this property.

Example

This example changes all corner nodes to smooth nodes in the third shape on the active document. The third shape must be a freeform drawing.

```
Dim docActive As Document
Dim intCount As Integer

Set docActive = ActiveDocument

With docActive.Shapes(3).Nodes
    For intCount = 1 to .Count
        If .Item(intCount).EditingType = msoEditingCorner Then
            .SetEditingType intCount, msoEditingSmooth
        End If
    Next
End With
```



Email Property

-

Returns an [Email](#) object that contains all the e-mail – related properties of the current document. Read-only.

Example

This example returns the name of the style associated with the current e-mail author.

```
MsgBox ActiveDocument.Email _  
    .CurrentEmailAuthor.Style.NameLocal
```



EmailOptions Property

-

Returns an [EmailOptions](#) object that represents the global preferences for e-mail authoring. Read-only.

Example

This example sets Microsoft Word to mark comments in e-mail messages.

```
Application.EmailOptions.MarkComments = True
```



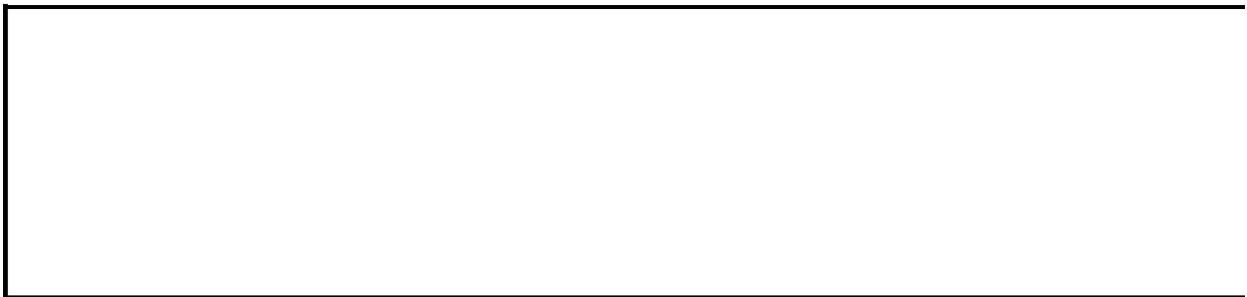
EmailSignature Property

Returns an [EmailSignature](#) object that represents the signatures Microsoft Word appends to outgoing e-mail messages. Read-only.

Example

This example displays the signature Word appends to new outgoing e-mail messages.

```
With Application.EmailOptions.EmailSignature
    If .NewMessageSignature = "" Then
        MsgBox "There is no signature for new " _
            & "e-mail messages!"
    Else
        MsgBox "The signature for new e-mail" _
            & "messages is: " & vbLf & vbLf _
            & .NewMessageSignature
    End If
End With
```



EmailSignatureEntries Property

Returns an [EmailSignatureEntries](#) object that represents the e-mail signature entries in Microsoft Word. Read-only.

expression.**EmailSignatureEntries**

expression Required. An expression that returns an [EmailSignature](#) object.

Remarks

An e-mail signature is standard text that ends an e-mail message, such as your name and telephone number. Use the **EmailSignatureEntries** property to create and manage a collection of e-mail signatures that Word will use when creating e-mail messages.

Example

This example creates a new signature entry based on the author's name and the selection in the active document.

```
Sub NewSignature()  
    Application.EmailOptions.EmailSignature _  
        .EmailSignatureEntries.Add _  
        Name:=ActiveDocument.BuiltInDocumentProperties("Author"), _  
        Range:=Selection.Range  
End Sub
```



EmailSubject Property

-

Returns or sets the text string for the specified hyperlink's subject line. The subject line is appended to the hyperlink's Internet address, or URL. Read/write **String**.

Remarks

This property is commonly used with e-mail hyperlinks. The value of this property takes precedence over any e-mail subject specified in the [Address](#) property of the same **Hyperlink** object.

Example

This example checks the active document for e-mail hyperlinks; if it finds any that have a blank subject line, it adds the subject "NewProducts".

```
Dim hypLoop As Hyperlink

For Each hypLoop In ActiveDocument.Hyperlinks
    If hypLoop.Address Like "mailto*" And _
        hypLoop.Address = hypLoop.EmailSubject Then
        hypLoop.EmailSubject = "NewProducts"
    End If
Next hypLoop
```



EmailTemplate Property

Returns or sets a **String** that represents the document template to use when sending e-mail messages. Read/write.

expression.**EmailTemplate**

expression Required. An expression that returns an [Application](#) object.

Remarks

Use the **EmailTemplate** property when Microsoft Word is specified as your e-mail editor, which you must do inside Microsoft Outlook.

Example

This example instructs Word to use the template named "Email" for all new e-mail messages. This example assumes that you have a template named "Email" and that it is stored in the default template location.

```
Sub MessageTemplate()  
    Application.EmailTemplate = "Email"  
End Sub
```



EmbedLinguisticData Property

-
True for Microsoft Word to embed speech and handwriting so that data can be converted back to speech or handwriting and to store East Asian IME keystrokes to improve correction; also controls text service data received from devices connected to Microsoft Office using the Windows Text Service Framework Application Programming Interface. Read/write **Boolean**.

expression.**EmbedLinguisticData**

expression Required. An expression that returns a [Document](#) object.

Example

This example embeds into the active document any speech or handwriting that may exist in the document.

```
Sub EmbedSpeechHandwriting()  
    ActiveDocument.EmbedLinguisticData = True  
End Sub
```



EmbedSmartTag Property

True for Microsoft Word to save the smart tag information in HTML e-mail messages. Read/write **Boolean**.

expression.**EmbedSmartTag**

expression Required. An expression that returns an [EmailOptions](#) object.

Remarks

Use the **EmbedSmartTag** property when Word is specified as your e-mail editor and messages are sent using HTML. This allows recipients of the message to have access to the smart tag information without having the recognizer file registered on their computer. To make Word your default e-mail editor, change the necessary settings in Microsoft Outlook.

Example

This example enables embedding smart tag information in e-mail messages. This example assumes that Word is your default e-mail editor.

```
Sub EmbedSmartTagsInEmail()  
    Application.EmailOptions.EmbedSmartTag = True  
End Sub
```



EmbedSmartTags Property

True for Microsoft Word to save the smart tag information in a document.
Read/write **Boolean**.

expression.**EmbedSmartTags**

expression Required. An expression that returns a [Document](#) object.

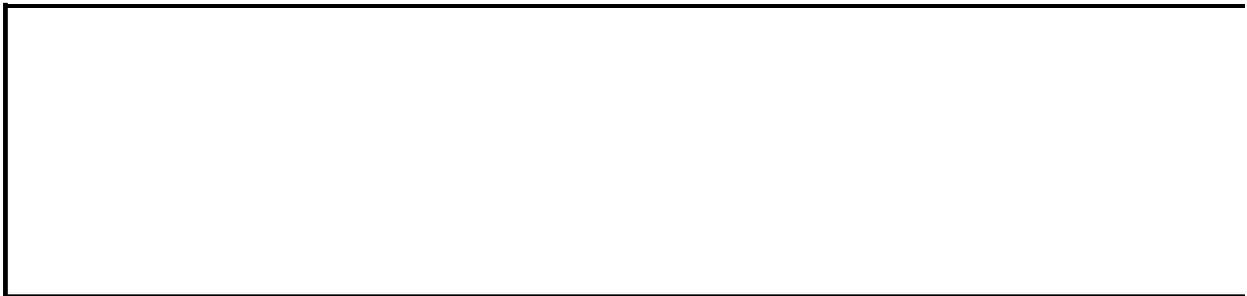
Remarks

Use the **EmbedSmartTags** property when sending documents to users who may not have the smart tag recognizer file on their computer. This allows the recipient to still have access to the smart tag information (and to the related actions if they have the smart tag actions file on their computer). However, if a document containing smart tags is edited by a user with an earlier version of Word, the smart tag information is removed.

Example

This example turns off saving smart tag information with the active document, which requires that recipients of the document have the necessary smart tag recognizer files registered on their computer and enabled through the **Smart Tags** tab of the **AutoCorrect** dialog.

```
Sub DontEmbedSmartTags()  
    ActiveDocument.EmbedSmartTags = False  
End Sub
```



EmbedTrueTypeFonts Property

-

True if Microsoft Word embeds TrueType fonts in a document when it's saved. This allow others to view the document with the same fonts that were used to create it. Read/write **Boolean**.

Example

This example sets Word to automatically embed TrueType fonts when saving a document, and then it saves the active document.

```
ActiveDocument.EmbedTrueTypeFonts = True  
ActiveDocument.Save
```

This example returns the current status of the **Embed TrueType fonts** check box in the **Save options** area on the **Save** tab in the **Options** dialog box.

```
temp = ActiveDocument.EmbedTrueTypeFonts
```



Emboss Property

-
True if the specified font is formatted as embossed. Returns **True**, **False**, or **wdUndefined**. Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

Remarks

Setting **Emboss** to **True** sets **Engrave** to **False**, and vice versa.

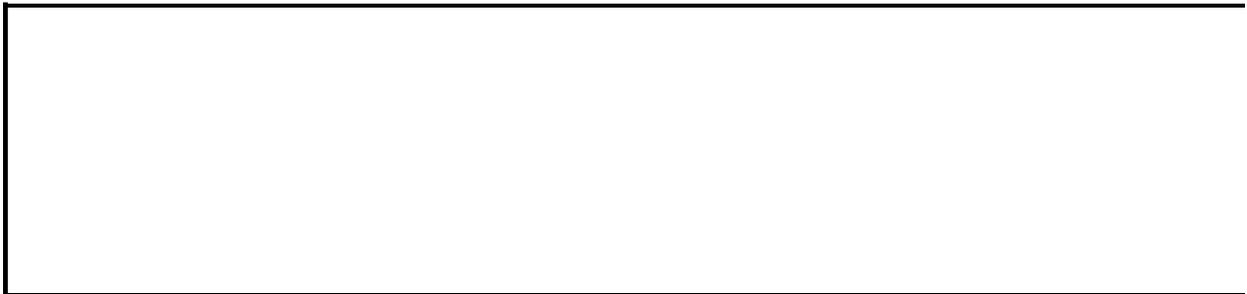
Example

This example embosses the second sentence in a new document.

```
With Documents.Add.Content
    .InsertAfter "This is the first sentence. "
    .InsertAfter "This is the second sentence. "
    .Sentences(2).Font.Emboss = True
End With
```

This example embosses the selected text.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Emboss = True
Else
    MsgBox "You need to select some text."
End If
```



EmphasisMark Property

Returns or sets the emphasis mark for a character or designated character string.
Read/write [WdEmphasisMark](#).

WdEmphasisMark can be one of these WdEmphasisMark constants.

wdEmphasisMarkNone

wdEmphasisMarkOverComma

wdEmphasisMarkOverSolidCircle

wdEmphasisMarkOverWhiteCircle

wdEmphasisMarkUnderSolidCircle

expression.**EmphasisMark**

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

Example

This example sets the emphasis mark over the fourth word in the active document to a comma.

```
ActiveDocument.Words(4).EmphasisMark = wdEmphasisMarkOverComma
```



Empty Property

-
True if the specified bookmark is empty. An empty bookmark marks a location (a collapsed selection); it doesn't mark any text. Read-only **Boolean**.

Note An error occurs if the specified bookmark doesn't exist. Use the **Exists** property to determine whether the bookmark exists.

Example

This example determines whether the bookmark named "temp" exists and whether it is empty.

```
If ActiveDocument.Bookmarks.Exists("temp") = True Then
    If ActiveDocument.Bookmarks("temp").Empty = True Then _
        MsgBox "The Temp bookmark is empty"
End If
```



Enable Property

-

Returns or sets border formatting for the specified object. Returns **True** or **wdUndefined** if border formatting is applied to all or part of the specified object. Can be set to **True**, **False**, or a [WdLineStyle](#) constant. Read/write **Long**.

Remarks

The **Enable** property applies to all borders for the specified object. **True** sets the line style to the default line style and sets the line width to the default line width. The default line style and line width can be set using the **DefaultBorderLineWidth** and **DefaultBorderLineStyle** properties.

To remove all the borders from an object, set the **Enable** property to **False**, as shown in the following example.

```
ActiveDocument.Tables(1).Borders.Enable = False
```

To remove or apply a single border, use **Borders(index)**, where *index* is a **WdBorderType** constant, to return a single border, and then set the **LineStyle** property. The following example removes the bottom border from rngTemp.

```
Dim rngTemp
```

```
rngTemp.Borders(wdBorderBottom).LineStyle = wdLineStyleNone
```

Example

This example removes all borders from the first cell in table one.

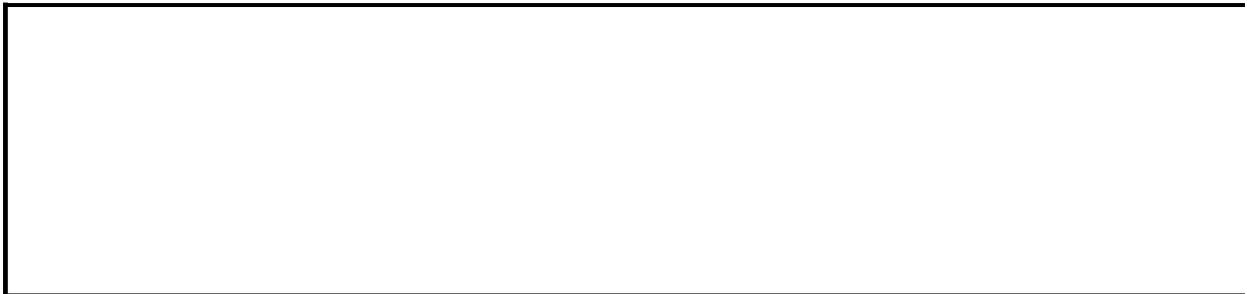
```
If ActiveDocument.Tables.Count >= 1 Then
    ActiveDocument.Tables(1).Cell(1, 1).Borders.Enable = False
End If
```

This example applies a dashed border around the first paragraph in the selection.

```
Options.DefaultBorderLineWidth = wdLineWidth025pt
Selection.Paragraphs(1).Borders.Enable = _
    wdLineStyleDashSmallGap
```

This example applies a border around the first character in the selection. If nothing is selected, the border is applied to the first character after the insertion point.

```
Selection.Characters(1).Borders.Enable = True
```



EnableCancelKey Property

Returns or sets the way that Word handles CTRL+BREAK user interruptions. Read/write [WdEnableCancelKey](#).

WdEnableCancelKey can be one of these WdEnableCancelKey constants.
wdCancelDisabled Prevents CTRL+BREAK from interrupting a macro.
wdCancelInterrupt Allows a macro to be interrupted by CTRL+BREAK.

expression.**EnableCancelKey**

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

Remarks

Use this property very carefully. If you use **wdCancelDisabled**, there's no way to interrupt a runaway loop or other non – self-terminating code. Also, the **EnableCancelKey** property is not reset to **wdCancelInterrupt** when your code stops running; unless you explicitly reset its value, it will remain set to **wdCancelDisabled** for the duration of the Word session.

Example

This example disables CTRL+BREAK from interrupting a counter loop.

```
Dim intWait As Integer
```

```
Application.EnableCancelKey = wdCancelDisabled
```

```
For intWait = 1 To 10000
```

```
    StatusBar = intWait
```

```
Next intWait
```

```
Application.EnableCancelKey = wdCancelInterrupt
```



Enabled Property

-
True if a form field is enabled. If a form field is enabled, its contents can be changed as the form is filled in. Read/write **Boolean**.

Example

If the first form field in the active document is an enabled check box, this example selects the check box.

```
Dim ffFirst As FormField

Set ffFirst = ActiveDocument.FormFields(1)
If ffFirst.Enabled = True And _
    ffFirst.Type = wdFieldFormCheckBox Then
    ffFirst.CheckBox.Value = True
End If
```



EnableFirstPageInSection Property

-
True if page borders are enabled for the first page in the section. Read/write **Boolean**.

Example

This example adds a border around the first page in the first section in the selection.

```
Dim borderLoop As Border
```

```
With Selection.Sections(1)
```

```
    .Borders.EnableFirstPageInSection = True
```

```
    .Borders.EnableOtherPagesInSection = False
```

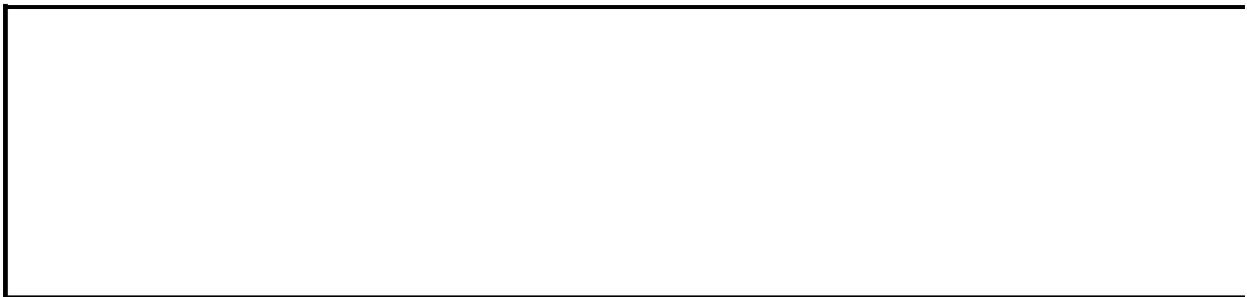
```
    For Each borderLoop In .Borders
```

```
        borderLoop.ArtStyle = wdArtPeople
```

```
        borderLoop.ArtWidth = 15
```

```
    Next borderLoop
```

```
End With
```



EnableHangulHanjaRecentOrdering Property

True if Microsoft Word displays the most recently used words at the top of the suggestions list during conversion between Hangul and Hanja. Read/write **Boolean**.

expression.**EnableHangulHanjaRecentOrdering**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example asks the user whether to set Microsoft Word to display the most recently used words at the top of the suggestions list during conversion between Hangul and Hanja.

```
x = MsgBox("Display most recently used words " _  
    & "at the top of the suggestions list?", vbYesNo)  
If x = vbYes Then  
    Options.EnableHangulHanjaRecentOrdering = True  
Else  
    Options.EnableHangulHanjaRecentOrdering = False  
End If
```



EnableMisusedWordsDictionary Property

-

True if Microsoft Word checks for misused words when checking the spelling and grammar in a document. Read/write **Boolean**.

Remarks

Word looks for the following when checking for misused words: incorrect usage of adjectives and adverbs, comparatives and superlatives, "like" as a conjunction, "nor" versus "or," "what" versus "which," "who" versus "whom," units of measurement, conjunctions, prepositions, and pronouns.

Example

This example sets Word to ignore misused words when checking spelling and grammar.

```
Options.EnableMisusedWordsDictionary = False
```



EnableOtherPagesInSection Property

-
True if page borders are enabled for all pages in the section except for the first page. Read/write **Boolean**.

Example

This example adds a border around each page in the first section in the selection except for the first page.

```
Dim borderLoop As Border
```

```
With Selection.Sections(1)
```

```
    .Borders.EnableFirstPageInSection = False
```

```
    .Borders.EnableOtherPagesInSection = True
```

```
    For Each borderLoop In .Borders
```

```
        borderLoop.ArtStyle = wdArtBabyRattle
```

```
        borderLoop.ArtWidth = 22
```

```
    Next borderLoop
```

```
End With
```



EnableSound Property

-

True if Word makes the computer respond with a sound whenever an error occurs. Read/write **Boolean**.

Example

This example sets the **Provide feedback with sound** option on the **General** tab in the **Options** dialog box, based on user input.

```
If MsgBox("Do you want Word to beep on errors?", 36) = vbYes Then
    Options.EnableSound = True
Else
    Options.EnableSound = False
End If
```



EnclosureNumber Property

-
Returns or sets the number of enclosures for a letter created by the Letter Wizard. Read/write **String**.

Example

This example displays the number of enclosures specified in the active document.

```
MsgBox ActiveDocument.GetLetterContent.EnclosureNumber
```

This example retrieves letter elements from the active document, changes the number of enclosures by setting the **EnclosureNumber** property, and then uses the **SetLetterContent** method to update the active document to reflect the changes.

```
Dim lcTemp As LetterContent
```

```
Set lcTemp = ActiveDocument.GetLetterContent
```

```
lcTemp.EnclosureNumber = "5"
```

```
ActiveDocument.SetLetterContent LetterContent:=lcTemp
```



Encoding Property

Returns or sets the document encoding (code page or character set) to be used by the Web browser when you view the saved document. Read/write [MsoEncoding](#).

MsoEncoding can be one of these MsoEncoding constants; however, you cannot use any of the constants that have the suffix **AutoDetect**. These constants are used by the [ReloadAs](#) method.

msoEncodingOEMMultilingualLatinI

msoEncodingOEMNordic

msoEncodingOEMTurkish

msoEncodingSimplifiedChineseAutoDetect

msoEncodingT61

msoEncodingTaiwanEten

msoEncodingTaiwanTCA

msoEncodingTaiwanWang

msoEncodingTraditionalChineseAutoDetect

msoEncodingTurkish

msoEncodingUnicodeLittleEndian

msoEncodingUTF7

msoEncodingVietnamese

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDICKoreanExtendedAndKorean

msoEncodingEBCDICMultilingualROECELatin2

msoEncodingEBCDICSerbianBulgarian

msoEncodingEBCDICThai

msoEncodingEBCDICTurkishLatin5

msoEncodingEBCDICUSCanada

msoEncodingEBCDICUSCanadaAndTraditionalChinese

msoEncodingOEMModernGreek
msoEncodingOEMMultilingualLatinII
msoEncodingOEMPortuguese
msoEncodingOEMUnitedStates
msoEncodingSimplifiedChineseGBK
msoEncodingTaiwanCNS
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTeleText
msoEncodingThai
msoEncodingTraditionalChineseBig5
msoEncodingUnicodeBigEndian
msoEncodingUSASCII
msoEncodingUTF8
msoEncodingWestern
msoEncodingArabic
msoEncodingArabicASMO
msoEncodingArabicAutoDetect
msoEncodingArabicTransparentASMO
msoEncodingAutoDetect
msoEncodingBaltic
msoEncodingCentralEuropean
msoEncodingCyrillic
msoEncodingCyrillicAutoDetect
msoEncodingEBCDICArabic
msoEncodingEBCDICDenmarkNorway
msoEncodingEBCDICFinlandSweden
msoEncodingEBCDICFrance
msoEncodingEBCDICGermany
msoEncodingEBCDIGreek
msoEncodingEBCDIGreekModern
msoEncodingEBCDICHebrew
msoEncodingEBCDICIcelandic
msoEncodingEBCDICInternational

msoEncodingEBCDICItaly
msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese
msoEncodingEBCDIKKoreanExtended
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICRussian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICTurkish
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
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

expression.**Encoding**

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

Example

This example checks to see whether the default document encoding is Western, and then it sets the string `strDocEncoding` accordingly.

```
Dim strDocEncoding As String

If Application.DefaultWebOptions.Encoding _
    = msoEncodingWestern Then
    strDocEncoding = "Western"
Else
    strDocEncoding = "Other"
End If
```



End Property

Returns or sets the ending character position of a selection, range, or bookmark.
Read/write **Long**.

Note If this property is set to a value smaller than the [Start](#) property, the **Start** property is set to the same value (that is, the **Start** and **End** property are equal).

Remarks

The **Selection**, **Range**, and **Bookmark** objects all have a starting position and an ending position. The ending position is the point farthest away from the beginning of the story.

This property returns the ending character position relative to the beginning of the story. The main document story (**wdMainTextStory**) begins with character position 0 (zero). You can change the size of a selection, range, or bookmark by setting this property.

Example

This example compares the ending position of the "temp" bookmark with the starting position of the "begin" bookmark.

```
Set Book1 = ActiveDocument.Bookmarks("begin")
Set Book2 = ActiveDocument.Bookmarks("temp")
If Book2.End > Book1.Start Then Book1.Select
```

This example retrieves the ending position of the selection. This value is used to create a range so that a field can be inserted after the selection.

```
pos = Selection.End
Set myRange = ActiveDocument.Range(Start:=pos, End:=pos)
ActiveDocument.Fields.Add Range:=myRange, Type:=wdFieldAuthor
```

This example changes the ending position of myRange by one character.

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.End = myRange.End - 1
```



EndArrowheadLength Property

Returns or sets the length of the arrowhead at the end of the specified line. Read/write [MsoArrowheadLength](#).

MsoArrowheadLength can be one of these MsoArrowheadLength constants.

msoArrowheadLengthMixed

msoArrowheadShort

msoArrowheadLengthMedium

msoArrowheadLong

expression.**EndArrowheadLength**

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

Example

This example adds a line to the active document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(100, 100, 200, 300).Line  
    .BeginArrowheadLength = msoArrowheadShort  
    .BeginArrowheadStyle = msoArrowheadOval  
    .BeginArrowheadWidth = msoArrowheadNarrow  
    .EndArrowheadLength = msoArrowheadLong  
    .EndArrowheadStyle = msoArrowheadTriangle  
    .EndArrowheadWidth = msoArrowheadWide  
End With
```



EndArrowheadStyle Property

Returns or sets the style of the arrowhead at the end of the specified line. Read/write [MsoArrowheadStyle](#).

MsoArrowheadStyle can be one of these MsoArrowheadStyle constants.

msoArrowheadNone

msoArrowheadOval

msoArrowheadStyleMixed

msoArrowheadDiamond

msoArrowheadOpen

msoArrowheadStealth

msoArrowheadTriangle

expression.EndArrowheadStyle

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

Example

This example adds a line to the active document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(100, 100, 200, 300).Line  
    .BeginArrowheadLength = msoArrowheadShort  
    .BeginArrowheadStyle = msoArrowheadOval  
    .BeginArrowheadWidth = msoArrowheadNarrow  
    .EndArrowheadLength = msoArrowheadLong  
    .EndArrowheadStyle = msoArrowheadTriangle  
    .EndArrowheadWidth = msoArrowheadWide  
End With
```



EndArrowheadWidth Property

Returns or sets the width of the arrowhead at the end of the specified line. Read/write [MsoArrowheadWidth](#).

MsoArrowheadWidth can be one of these MsoArrowheadWidth constants.

msoArrowheadNarrow

msoArrowheadWidthMedium

msoArrowheadWide

msoArrowheadWidthMixed

expression.**EndArrowheadWidth**

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

Example

This example adds a line to the active document. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes.AddLine(100, 100, 200, 300).Line  
    .BeginArrowheadLength = msoArrowheadShort  
    .BeginArrowheadStyle = msoArrowheadOval  
    .BeginArrowheadWidth = msoArrowheadNarrow  
    .EndArrowheadLength = msoArrowheadLong  
    .EndArrowheadStyle = msoArrowheadTriangle  
    .EndArrowheadWidth = msoArrowheadWide  
End With
```



EndnoteOptions Property

Returns an [EndnoteOptions](#) object that represents the endnotes in a range or selection.

expression.**EndnoteOptions**

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

Example

This example sets the starting number for endnotes in section two of the active document to one if the starting number is not one.

```
Sub SetEndnoteOptionsRange()  
    With ActiveDocument.Sections(2).Range.EndnoteOptions  
        If .StartingNumber <> 1 Then  
            .StartingNumber = 1  
        End If  
    End With  
End Sub
```



Endnotes Property

-
Returns an [Endnotes](#) collection that represents all the endnotes in a range, selection, or document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example positions the endnotes in the active document at the end of the document and formats the endnote reference marks as lowercase roman numerals.

```
With ActiveDocument.Endnotes  
    .Location = wdEndOfDocument  
    .NumberStyle = wdNoteNumberStyleLowercaseRoman  
End With
```



Engrave Property

-

True if the font is formatted as engraved. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**.
Read/write **Long**.

Remarks

Setting **Engrave** to **True** sets **Emboss** to **False**, and vice versa.

Example

This example formats the first letter in the active document as engraved.

```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Characters(1)
With rngTemp.Font
    .Size = 20
    .Engrave = True
End With
```

This example formats the selection as engraved.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Engrave = True
Else
    MsgBox "You need to select some text."
End If
```



Entries Property

Returns an [AutoCorrectEntries](#) collection that represents the current list of AutoCorrect entries. This list corresponds to the list of AutoCorrect entries on the **AutoCorrect** tab in the **AutoCorrect** dialog box (**Tools** menu). Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the total number of AutoCorrect entries.

```
MsgBox AutoCorrect.Entries.Count
```

This example deletes the specified AutoCorrect entry if it exists.

```
Dim strEntry As String
Dim acEntry As AutoCorrectEntry
Dim blnMatch As Boolean
Dim intResponse As Integer

strEntry = InputBox("Enter the AutoCorrect entry to delete.")
blnMatch = False

For Each acEntry in AutoCorrect.Entries
    If acEntry.Name = strEntry Then
        blnMatch = True
        intResponse = _
            MsgBox("Are you sure you want to delete " _
                & acEntry.Name, 4)
        If intResponse = vbYes Then
            acEntry.Delete
        End If
    End If
Next acEntry

If blnMatch <> True Then
    MsgBox "There was no AutoCorrect entry: " & strEntry
End If
```



EntryMacro Property

-

Returns or sets an entry macro name for the specified form field (**CheckBox**, **DropDown**, or **TextInput**). The entry macro runs when the form field gets the focus. Read/write **String**.

Example

This example assigns the macro named "Blue" to the first form field in "Form.doc."

```
Documents("Form.doc").FormFields(1).EntryMacro = "Blue"
```

This example assigns the macro named "Breadth" to the form field named "Text1" in the active document.

```
ActiveDocument.FormFields("Text1").EntryMacro = "Breadth"
```



EntrySeparator Property

-

Returns or sets the characters (up to five) that separate a table of authorities entry and its page number. The default is a tab character with a dotted leader.

Corresponds to the \e switch for a TOA (Table of Authorities) field. Read/write **String**.

Example

This example inserts a table of authorities into the active document and then formats the table to use a comma between the entries and their corresponding page numbers.

```
Dim rngTemp As Range
Dim toaLoop As TableOfAuthorities

Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
ActiveDocument.TablesOfAuthorities.Add _
    Range:=rngTemp, Category:=1
For Each toaLoop In ActiveDocument.TablesOfAuthorities
    toaLoop.EntrySeparator = ", "
Next toaLoop
```

This example returns the entry separator for the first table of authorities.

```
Dim strSeparator

strSeparator = _
    ActiveDocument.TablesOfAuthorities(1).EntrySeparator
```



Envelope Property

-

Returns an [Envelope](#) object that represents envelope functionality and the envelope in the specified document. Read-only.

Example

This example sets the default envelope size to C4 (229 x 324 mm).

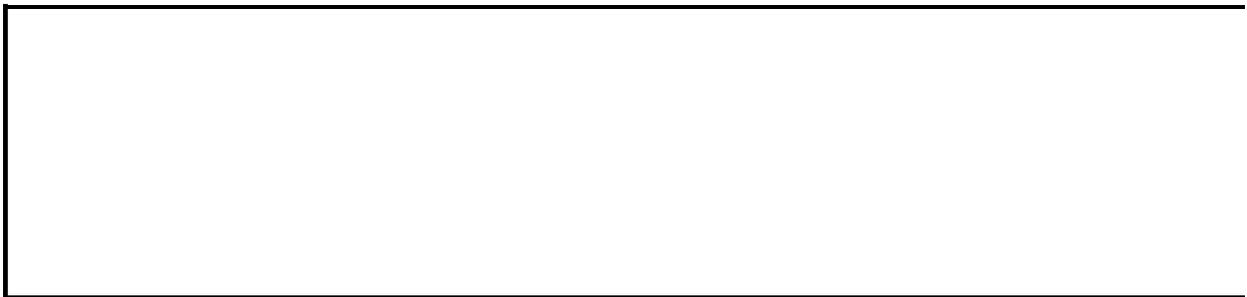
```
ActiveDocument.Envelope.DefaultSize = "C4"
```

This example displays the delivery address if an envelope has been added to the document; otherwise, a message box is displayed.

```
On Error GoTo errhandler  
addr = ActiveDocument.Envelope.Address.Text  
MsgBox Prompt:=addr, Title:="Delivery Address"  
errhandler:  
If Err = 5852 Then MsgBox "Add an envelope to the document"
```

This example creates a new document and adds an envelope with a predefined delivery address and return address.

```
addr = "Don Funk" & vbCr & "123 Skye St." _  
      & vbCr & "Our Town, WA 98040"  
retaddr = "Karin Gallagher" & vbCr & "123 Main" _  
         & vbCr & "Other Town, WA 98004"  
Documents.Add.Envelope.Insert Address:=addr, ReturnAddress:=retaddr  
ActiveDocument.ActiveWindow.View.Type = wdPrintView
```



EnvelopeFeederInstalled Property

-
True if the current printer has a special feeder for envelopes. Read-only
Boolean.

Example

This example prints the active document as an envelope, provided that there's an envelope feeder installed.

```
If Options.EnvelopeFeederInstalled = True Then
    ActiveDocument.Envelope.PrintOut _
        AddressFromLeft:=InchesToPoints(3), _
        AddressFromTop:=InchesToPoints(1.5)
Else
    MsgBox "No envelope feeder available."
End If
```



EnvelopeVisible Property

-

True if the e-mail message header is visible in the document window. The default value is **False**. This property has no effect if the document isn't an e-mail message. Read/write **Boolean**.

Example

This example displays the e-mail message header.

```
ActiveWindow.EnvelopeVisible = True
```



EvenlySpaced Property

-

True if text columns are evenly spaced. Can be **True**, **False**, or **wdUndefined**.
Read/write **Long**.

Remarks

If you set the **Spacing** or **Width** property of the **TextColumns** object, the **EvenlySpaced** property is automatically set to **True**. Also, setting the **EvenlySpaced** property may change the settings for the **Spacing** and **Width** properties of the **TextColumns** object.

Example

This example topic sets columns in the active document to be evenly spaced.

```
Dim colTextColumns

Set colTextColumns = ActiveDocument.PageSetup.TextColumns

If colTextColumns.Count > 1 Then _
    colTextColumns.EvenlySpaced = True
End If
```

This example returns the status of the **Equal column width** option in the **Columns** dialog box (**F**ormat menu).

```
Dim lngSpaced As Long

lngSpaced = ActiveDocument.PageSetup.TextColumns.EvenlySpaced
```



Exists Property

True if the specified **HeaderFooter** object exists. Read/write **Boolean**.

Note The primary header and footer exist in all new documents by default. Use this method to determine whether a first-page or odd-page header or footer exists. You can also use the [DifferentFirstPageHeaderFooter](#) or [OddAndEvenPagesHeaderFooter](#) property to return or set the number of headers and footers in the specified document or section.

Example

If a first-page header exists in section one, this example sets the text for the header.

```
Dim secTemp As Section
```

```
Set secTemp = ActiveDocument.Sections(1)
```

```
If secTemp.Headers(wdHeaderFooterFirstPage).Exists = True Then  
    secTemp.Headers(wdHeaderFooterFirstPage).Range.Text = _  
        "First Page"
```

```
End If
```



ExitMacro Property

-

Returns or sets an exit macro name for the specified form field (**CheckBox**, **DropDown**, or **TextInput**). The exit macro runs when the form field loses the focus. Read/write **String**.

Example

This example assigns the macro named "Reformat" to the first form field in the selection.

```
If Selection.FormFields.Count > 0 Then _  
    Selection.FormFields(1).ExitMacro = "Reformat"
```

This example assigns the macro named "Blue" to the last form field in "Form.doc."

```
Dim intMax As Integer
```

```
intMax = Documents("Form.doc").FormFields.Count  
Documents("Form.doc").FormFields(intMax).ExitMacro = "Blue"
```



Expanded Property

-

True if the subdocuments in the specified document are expanded. Read/write
Boolean.

Example

This example expands all subdocuments in the active master document.

```
If ActiveDocument.Subdocuments.Count >= 1 Then
    ActiveDocument.Subdocuments.Expanded = True
End If
```

This example toggles the **Expanded** property between expanding and collapsing all subdocuments in the active document.

```
ActiveDocument.Subdocuments.Expanded = _
    Not ActiveDocument.Subdocuments.Expanded
```

This example determines whether the subdocuments in Report.doc are expanded and then displays a message indicating their status.

```
If Documents("Report.doc").Subdocuments.Expanded = True Then
    MsgBox "All available information is displayed."
Else
    MsgBox "Expand subdocuments for more information."
End If
```



ExtendMode Property

True if Extend mode is active. When Extend mode is active, the *Extend* argument of the following methods is **True** by default: [EndKey](#), [HomeKey](#), [MoveDown](#), [MoveLeft](#), [MoveRight](#), and [MoveUp](#). Also, the letters "EXT" appear on the status bar. Read/write **Boolean**.

expression.**ExtendMode**

expression Required. An expression that returns a [Selection](#) object.

Remarks

This property can only be set during run time; attempts to set it in Immediate mode are ignored. The *Extend* arguments of the [EndOf](#) and [StartOf](#) methods are not affected by this property.

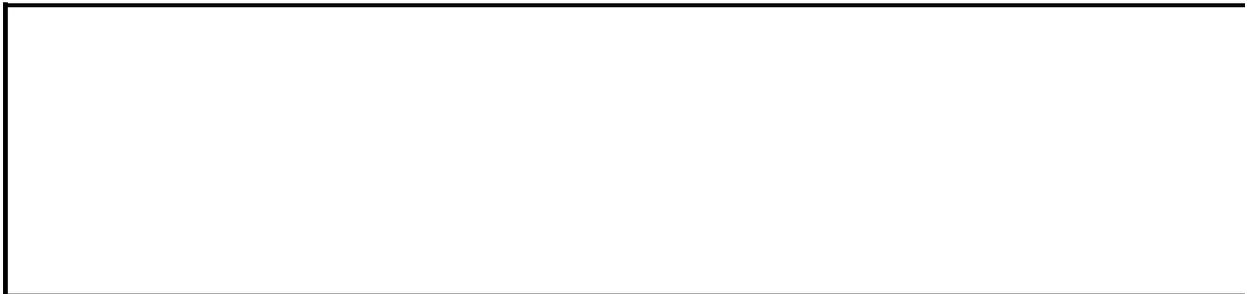
Example

This example moves to the beginning of the paragraph and selects the paragraph plus the next two sentences.

```
With Selection
    .MoveUp Unit:=wdParagraph
    .ExtendMode = True
    .MoveDown Unit:=wdParagraph
    .MoveRight Unit:=wdSentence, Count:=2
End With
```

This example collapses the current selection, turns on Extend mode, and selects the current sentence.

```
With Selection
    .Collapse
    .ExtendMode = True
    ' Select current word.
    .Extend
    ' Select current sentence.
    .Extend
End With
```



Extensions Property

-

Returns the file name extensions associated with the specified [FileConverter](#) object. Read-only **String**.

Example

This example displays the name and file name extensions for first file converter.

```
Dim fcTemp As FileConverter
```

```
Set fcTemp = FileConverters(1)
```

```
MsgBox "The file extensions for " & fcTemp.FormatName _  
    & " files are: " & fcTemp.Extensions
```



ExtraInfoRequired Property

-
True if extra information is required to resolve the specified hyperlink. Read-only **Boolean**.

Note You can specify extra information by using the *ExtraInfo* argument with the [Follow](#) or [FollowHyperlink](#) method. For example, you can use *ExtraInfo* to specify the coordinates of an image map, the contents of a form, or a FAT file name.

Example

This example inserts a hyperlink to www.msn.com and then follows the hyperlink if extra information isn't required.

```
Dim hypTemp As Hyperlink

With Selection
    .Collapse Direction:=wdCollapseEnd
    .InsertAfter "MSN "
    .Previous
End With
Set hypTemp = ActiveDocument.Hyperlinks.Add( _
    Address:="http://www.msn.com", _
    Anchor:=Selection.Range)
If hypTemp.ExtraInfoRequired = False Then
    hypTemp.Follow
End If
```



ExtrusionColor Property

-

Returns a [ColorFormat](#) object that represents the color of the shape's extrusion.
Read-only.

Example

This example adds an oval to the active document and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Dim docActive As Document
Dim shapeNew As Shape

Set docActive = ActiveDocument
Set shapeNew = docActive.Shapes.AddShape(msoShapeOval, _
    90, 90, 90, 40)
With shapeNew.ThreeD
    .Visible = True
    .Depth = 50
    ' RGB value for purple
    .ExtrusionColor.RGB = RGB(255, 100, 255)
End With
```



ExtrusionColorType Property

Returns or sets a value that indicates 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](#).

MsoExtrusionColorType can be one of these MsoExtrusionColorType constants.

msoExtrusionColorAutomatic Extrusion color based on shape fill.

msoExtrusionColorTypeMixed

msoExtrusionColorCustom Extrusion color independent of shape fill.

expression.**ExtrusionColorType**

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

Example

If the first shape on the active document has an automatic extrusion color, this example gives the extrusion a custom yellow color.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes(1).ThreeD
```

```
    If .ExtrusionColorType = msoExtrusionColorAutomatic Then
```

```
        .ExtrusionColor.RGB = RGB(240, 235, 16)
```

```
    End If
```

```
End With
```



FarEastLineBreakControl Property

-

True if Microsoft Word applies East Asian line-breaking rules to the specified paragraphs. Returns **wdUndefined** if the **FarEastLineBreakControl** property is set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example sets Word to apply East Asian line-breaking rules to the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).FarEastLineBreakControl = True
```



↳ [Show All](#)

FarEastLineBreakLanguage Property

Returns or sets the East Asian language to use when breaking lines of text in the specified document or template. Read/write [WdFarEastLineBreakLanguageID](#).

WdFarEastLineBreakLanguageID can be one of these WdFarEastLineBreakLanguageID constants.

wdLineBreakJapanese

wdLineBreakKorean

wdLineBreakSimplifiedChinese

wdLineBreakTraditionalChinese

reexpression.FarEastLineBakLanguage

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

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Word to break lines in the current document based on Korean language rules.

```
ActiveDocument.FarEastLineBreakLanguage = wdLineBreakKorean
```



↳ [Show All](#)

FarEastLineBreakLevel Property

Returns or sets the line break control level for the specified document. This property is ignored if the [FarEastLineBreakControl](#) property is set to **False**.
Read/write [WdFarEastLineBreakLevel](#).

WdFarEastLineBreakLevel can be one of these WdFarEastLineBreakLevel constants.

wdFarEastLineBreakLevelCustom

wdFarEastLineBreakLevelNormal

wdFarEastLineBreakLevelStrict

expression.**FarEastLineBreakLevel**

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

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Microsoft Word to perform line breaking on first-level *kinsoku* characters in the active document.

```
ActiveDocument.FarEastLineBreakLevel = wdJustificationModeCompressKa
```



FeatureInstall Property

Returns or sets how Microsoft Word handles calls to methods and properties that require features not yet installed. Read/write [MsoReatureInstall](#).

Can be one of the following MsoFeatureInstall constants.

Constant	Value	Description
msoFeatureInstallNone	0	The default value. A generic Automation error is generated at run time when uninstalled features are called.
msoFeatureInstallOnDemand	1	The user is prompted to install new features. A progress meter is displayed during
msoFeatureInstallOnDemandWithUI	2	installation. The user isn't prompted to install new features.

expression.**FeatureInstall**

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

Remarks

You can use the **msoFeatureInstallOnDemandWithUI** constant to prevent users from believing that the application isn't responding while a feature is being installed. Use the **msoFeatureInstallNone** constant if you want the developer to be the only one who can install features.

If you have the [DisplayAlerts](#) property set to **False**, users will not be prompted to install new features even if the **FeatureInstall** property is set to **msoFeatureInstallOnDemand**. If the **DisplayAlerts** property is set to **True**, an installation progress meter will appear if the **FeatureInstall** property is set to **msoFeatureInstallOnDemand**.

Example

This example activates a new instance of Microsoft Excel and checks the value of the **FeatureInstall** property. If the property is set to **msoFeatureInstallNone**, the code displays a message box that asks the user whether they want to change the property setting. If the user responds "Yes," the property is set to **msoFeatureInstallOnDemand**. For this example to function properly, you must add a reference to Microsoft Excel Object Library in the **References** dialog (**Tools** menu).

```
Dim ExcelApp As New Excel.Application
Dim intReply As Integer

With ExcelApp
    If .FeatureInstall = msoFeatureInstallNone Then
        intReply = MsgBox("Uninstalled features for " _
            & "this application may " & vbCrLf _
            & "cause a run-time error when called." _
            & vbCrLf & vbCrLf _
            & "Would you like to change this setting" & vbCrLf _
            & "to automatically install missing features?", _
            vbYesNo, "Feature Install Setting")
        If intReply = vbYes Then
            .FeatureInstall = msoFeatureInstallOnDemand
        End If
    End If
End With
```



FeedSource Property

Returns or sets the paper tray for the envelope. Read/write [WdPaperTray](#).

WdPaperTray can be one of these WdPaperTray constants.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormatBin

wdPrinterTractorFeed

wdPrinterUpperBin

expression.**FeedSource**

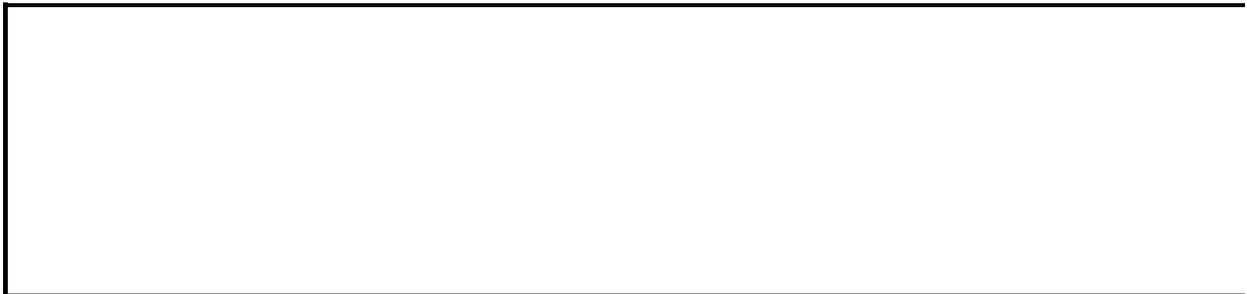
expression Required. An expression that returns one of the objects in the Applies To list.

Note If you use this property before an envelope has been added to the document, an error occurs.

Example

This example asks the user whether envelopes are fed into the printer manually. If the answer is yes, the example sets the paper tray to manual envelope feed.

```
Sub exFeedSource()  
  
    Dim intResponse As Integer  
  
    intResponse = _  
        MsgBox("Are the envelopes manually fed?", vbYesNo)  
    If intResponse = vbYes then  
        On Error GoTo errhandler  
        ActiveDocument.Envelope.FeedSource = _  
            wdPrinterManualEnvelopeFeed  
    End If  
  
    Exit Sub  
  
errhandler:  
    If Err = 5852 Then MsgBox _  
        "Envelope not part of the active document"  
  
End Sub
```



Field Property

-
Returns a [Field](#) object that represents the field associated with the specified shape. Read-only.

Note Use the [Fields](#) property to return the **Fields** collection.

Example

This example inserts a graphic as an inline shape (using an INCLUDEPICTURE field) and then displays the shape's field code.

```
Dim iShapeNew As InlineShape
```

```
Set iShapeNew = _  
    ActiveDocument.InlineShapes _  
        .AddPicture(FileName:="C:\Windows\Tiles.bmp", _  
            LinkToFile:=True, SaveWithDocument:=False, _  
            Range:=Selection.Range)
```

```
MsgBox iShapeNew.Field.Code.Text
```



FieldNames Property

Returns a [MailMergeFieldNames](#) collection that represents the names of all the fields in the specified mail merge data source. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the name of the first field in the data source attached to the active mail merge main document.

```
MsgBox ActiveDocument.MailMerge.DataSource.FieldNames(1).Name
```

This example uses the `mNames()` array to store the names of each merge field contained in the data source attached to the active document.

```
Dim mNames As Variant
Dim mmTemp As MailMerge
Dim intCount As Integer
Dim intIncrement As Integer
Dim mmfnLoop As MailMergeFieldName

Set mmTemp = ActiveDocument.MailMerge
intCount = _
    ActiveDocument.MailMerge.DataSource.FieldNames.Count - 1

ReDim mNames(intCount)
intIncrement = 0

For Each mmfnLoop In mmTemp.DataSource.FieldNames
    mNames(intIncrement) = mmfnLoop.Name
    intIncrement = intIncrement + 1
Next mmfnLoop
```



↳ [Show All](#)

Fields Property

▶ [Fields property as it applies to the **Document**, **Range**, and **Selection** objects.](#)

Returns a read-only **Fields** collection that represents all the fields in the document, range, or selection.

expression.**Fields**

expression Required. An expression that returns one of the above objects.

Note When applied to the **Document** object, the **Fields** property returns a **Fields** collection that contains only the fields in the main text story.

▶ [Fields property as it applies to the **MailMerge** object.](#)

Returns a read-only **MailMergeFields** collection that represents all the mail merge related fields in the specified document.

expression.**Fields**

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

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

▶ [As it applies to the Document, Range, and Selection objects.](#)

This example updates all the fields in the active document.

```
ActiveDocument.Fields.Update
```

This example removes all the fields from the main text story and the footer in the active document.

```
For Each aField in ActiveDocument.Fields
    aField.Delete
Next aField
Set myRange = ActiveDocument.Sections(1).Footers _
    (wdHeaderFooterPrimary).Range
For Each aField In myRange.Fields
    aField.Delete
Next aField
```

This example adds a DATE field at the insertion point.

```
With Selection
    .Collapse Direction:=wdCollapseStart
    .Fields.Add Range:=Selection.Range, Type:=wdFieldDate
End With
```

▶ [As it applies to the Document, Range, and Selection objects.](#)

This example adds a mail merge field named "Title" at the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart
ActiveDocument.MailMerge.Fields.Add Range:= Selection.Range, _
    Name:= "Title"
```



FieldShading Property

Returns or sets on-screen shading for form fields. Read/write [WdFieldShading](#).

WdFieldShading can be one of these WdFieldShading constants.

wdFieldShadingAlways

wdFieldShadingNever

wdFieldShadingWhenSelected

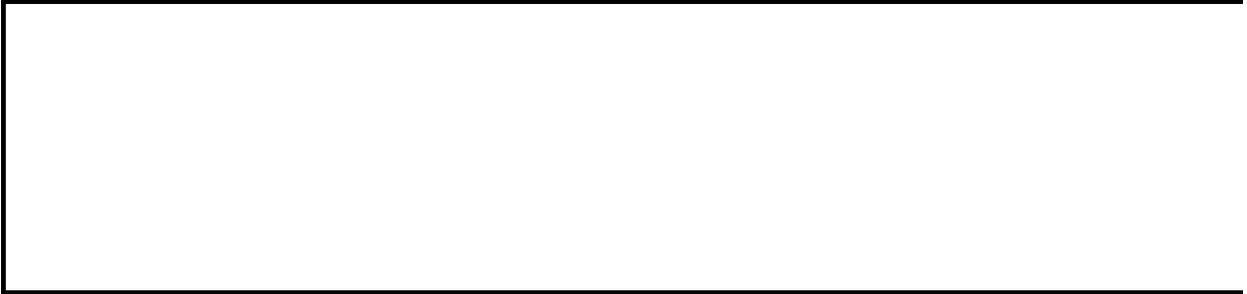
expression.**FieldShading**

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

Example

This example enables field shading for all form fields in the active window.

```
ActiveDocument.ActiveWindow.View.FieldShading = _  
    wdFieldShadingAlways
```



FileConverters Property

Returns a [FileConverters](#) collection that represents all the file converters available to Word. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the path of the WordPerfect 5.0 file converter.

```
MsgBox FileConverters("WrdPrfctDOS50").Path
```

This example displays a message that indicates whether the third converter in the **FileConverters** collection can save files.

```
If FileConverters(3).CanSave = True Then  
    MsgBox FileConverters(3).FormatName & " can save files"  
Else  
    MsgBox FileConverters(3).FormatName & " cannot save files"  
End If
```

This example displays the name of the last file converter.

```
Dim fcTemp As FileConverter  
  
Set fcTemp = FileConverters(FileConverters.Count)  
MsgBox "The file extensions for " & fcTemp.FormatName & _  
    " files are: " & fcTemp.Extensions
```



↳ [Show All](#)

FileDialog Property

Returns a **FileDialog** object which represents a single instance of a file dialog box.

expression.**FileDialog**(*FileDialogType*)

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

FileDialogType 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.

```
Sub ShowSaveAsDialog()  
    Dim dlgSaveAs As FileDialog  
    Set dlgSaveAs = Application.FileDialog( _  
        FileDialogType:=msoFileDialogSaveAs)  
    dlgSaveAs.Show  
End Sub
```

This example displays the **Open** dialog box and allows a user to select multiple files to open.

```
Sub ShowFileDialog()  
    Dim dlgOpen As FileDialog  
    Set dlgOpen = Application.FileDialog( _  
        FileDialogType:=msoFileDialogOpen)  
    With dlgOpen  
        .AllowMultiSelect = True  
        .Show  
    End With  
End Sub
```



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 an [Application](#) object.

Example

This example displays, in a series of message boxes, the file names in the My Documents folder that begin with 99.

```
With Application.FileSearch
    .FileName = "99*.*"
    .LookIn = "C:\My Documents"
    .Execute
    For I = 1 to .FoundFiles.Count
        MsgBox .FoundFiles(I)
    Next I
End With
```



Fill Property

-

Returns a [FillFormat](#) object that contains fill formatting properties for the specified shape. Read-only.

Example

This example adds a rectangle to myDocument and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
Set myDocument = Documents(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    90, 90, 90, 50).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(170, 170, 170)
    .TwoColorGradient msoGradientHorizontal, 1
End With
```



Filter Property

Returns or sets a value that specifies how Microsoft Word classifies the first character of entries in the specified index. Can be any of the following **WdIndexFilter** constants: **wdIndexFilterAiueo**, **wdIndexFilterAkasatana**, **wdIndexFilterChosung**, **wdIndexFilterLow**, **wdIndexFilterMedium**, **wdIndexFilterFull**, or **wdIndexFilterNone**. Read/write **Long**.

Example

This example inserts an index at the end of the active document. right-aligns the page numbers, and then sets Microsoft Word to classify index entries as "あかさたな".

```
Set myRange = ActiveDocument.Range _  
    (Start:=ActiveDocument.Content.End -1, _  
    End:=ActiveDocument.Content.End -1)  
ActiveDocument.Indexes.Add(Range:=myRange, Type:=wdIndexIndent, _  
    RightAlignPageNumbers:=True).Filter = wdIndexFilterAkasatana
```



Find Property

Returns a [Find](#) object that contains the criteria for a find operation. Read-only.

Note When this property is used with a **Selection** object, the selection is changed if the find operation is successful. If this property is used with a **Range** object, the selection isn't changed unless the **Select** method is applied.

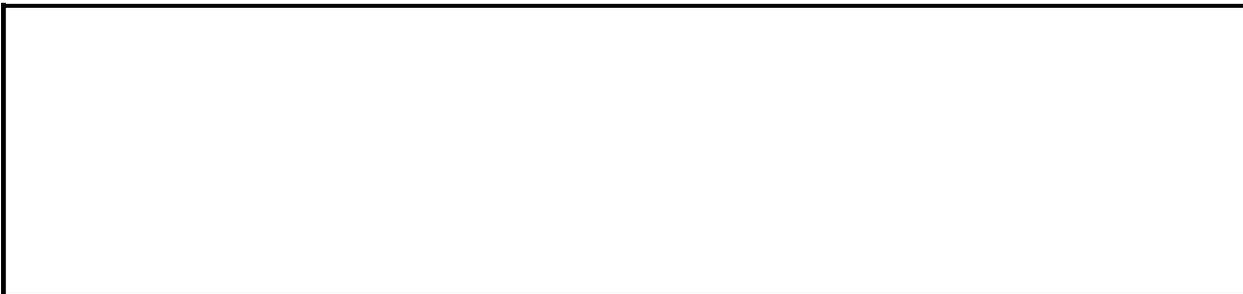
Example

The following example searches forward through the document for the word "Microsoft." If the word is found, it's automatically selected.

```
With Selection.Find
    .Forward = True
    .ClearFormatting
    .MatchWholeWord = True
    .MatchCase = False
    .Wrap = wdFindContinue
    .Execute FindText:="Microsoft"
End With
```

This example inserts "Tip: " at the beginning of every paragraph formatted with the Heading 3 style in the active document. The **Do...Loop** statement is used to repeat a series of actions each time this style is found.

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Style = wdStyleHeading3
    Do While .Execute(FindText:="", Forward:=True, _
        Format:=True) = True
        With .Parent
            .StartOf Unit:=wdParagraph, Extend:=wdMove
            .InsertAfter "Tip: "
            .Move Unit:=wdParagraph, Count:=1
        End With
    Loop
End With
```



FindKey Property

Returns a [KeyBinding](#) object that represents the specified key combination.
Read-only.

expression.**FindKey**(*KeyCode*, *KeyCode2*)

expression Optional. An expression that returns an **Application** object.

KeyCode Required **Long**. A key you specify by using one of the **WdKey** constants.

KeyCode2 Optional **Variant**. A second key you specify by using one of the **WdKey** constants.

Remarks

You can use the [BuildKeyCode](#) method to create the *KeyCode* or *KeyCode2* argument.

Example

This example disables the ALT+SHIFT+F12 key combination in the template attached to the active document. To return a **KeyBinding** object that includes more than two keys, use the **BuildKeyCode** method, as shown in the example.

```
CustomizationContext = ActiveDocument.AttachedTemplate  
FindKey(KeyCode:=BuildKeyCode(wdKeyAlt, wdKeyShift, _  
    wdKeyF12)).Disable
```

This example displays the command assigned to the F1 key.

```
CustomizationContext = NormalTemplate  
MsgBox FindKey(KeyCode:=wdKeyF1).Command
```



↳ [Show All](#)

First Property

▶ [First property as it applies to the **Characters**, **Sentences**, and **Words** objects.](#)

Returns a **Range** object that represents the first sentence, word, or character in a document, selection or range.

expression.**First**

expression Required. An expression that returns one of the above objects.

▶ [First property as it applies to the **Columns** object.](#)

Returns a **Column** object that represents the first item in the **Columns** collection.

expression.**First**

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

▶ [First property as it applies to the **Paragraphs** object.](#)

Returns a **Paragraph** object that represents the first item in the **Paragraphs** collection.

expression.**First**

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

▶ [First property as it applies to the **Rows** object.](#)

Returns a **Row** object that represents the first item in the **Rows** collection.

expression.**First**

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

▶ [First property as it applies to the **Sections** object.](#)

Returns a **Section** object that represents the first item in the **Sections** collection.

expression.**First**

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

Example

▶ [As it applies to the **Paragraph** object.](#)

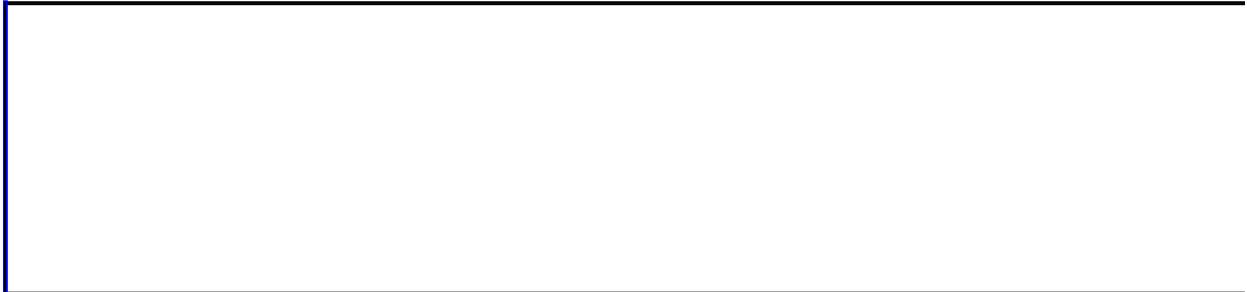
This example right-aligns the first paragraph in the selection.

```
Selection.Paragraphs.First.Alignment = wdAlignParagraphRight
```

▶ [As it applies to the **Rows** object.](#)

This example applies shading and a bottom border to the first row in the first table of the active document.

```
ActiveDocument.Tables(1).Borders.Enable = False  
With ActiveDocument.Tables(1).Rows.First  
    .Shading.Texture = wdTexture10Percent  
    .Borders(wdBorderBottom).LineStyle = wdLineStyleSingle  
End With
```



FirstChild Property

Returns a **DiagramNode** object that represents the first child node of a parent node. Read-only.

expression.**FirstChild**

expression Required. An expression that returns a [DiagramNodeChildren](#) object.

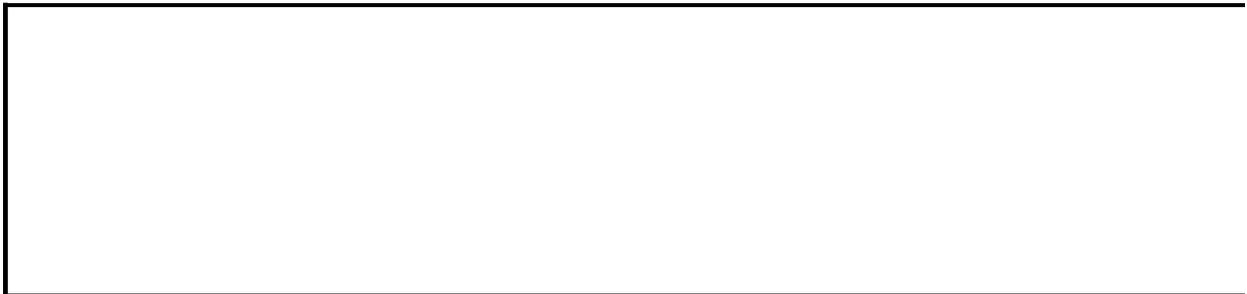
Remarks

Use the [LastChild](#) property to access the last child node. Use the [Root](#) property to access the parent node in a diagram.

Example

This example adds an organization chart diagram to the current document, adds three nodes, and assigns the first and last child nodes to variables.

```
Sub FirstChild()  
    Dim shpDiagram As Shape  
    Dim dgnRoot As DiagramNode  
    Dim dgnFirstChild As DiagramNode  
    Dim dgnLastChild As DiagramNode  
    Dim intCount As Integer  
  
    'Add organizational chart diagram to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
  
    'Add the first node to the diagram  
    Set dgnRoot = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next intCount  
  
    'Assign the first and last child nodes to variables  
    Set dgnFirstChild = dgnRoot.Children.FirstChild  
    Set dgnLastChild = dgnRoot.Children.LastChild  
End Sub
```



FirstLetterAutoAdd Property

-

True if Word automatically adds abbreviations to the list of AutoCorrect First Letter exceptions. Word adds an abbreviation to this list if you delete and then retype the letter that Word capitalized immediately after the period following the abbreviation. Read/write **Boolean**.

Example

This example prevents Word from automatically adding abbreviations to the list of AutoCorrect First Letter exceptions.

```
AutoCorrect.FirstLetterAutoAdd = False
```



FirstLetterExceptions Property

Returns a [FirstLetterExceptions](#) collection that represents the list of abbreviations after which Word won't automatically capitalize the next letter. This list corresponds to the list of AutoCorrect exceptions on the **First Letter** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu). Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds "apt." to the list of AutoCorrect First Letter exceptions.

```
AutoCorrect.FirstLetterExceptions.Add "apt."
```

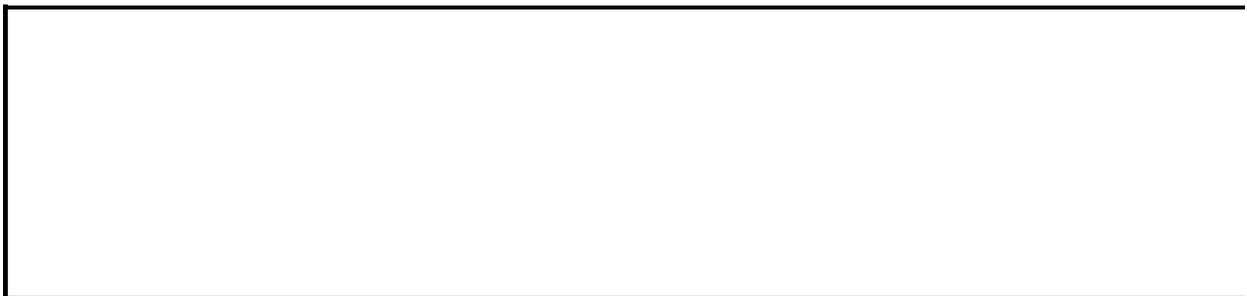
This example deletes the specified AutoCorrect First Letter exception if it exists.

```
Dim strException As String
Dim fleLoop As FirstLetterException
Dim blnMatch As Boolean
Dim intConfirm As Integer

strException = _
    InputBox("Enter the First Letter exception to delete.")
blnMatch = False

For Each fleLoop in AutoCorrect.FirstLetterExceptions
    If fleLoop.Name = strException Then
        blnMatch = True
        intConfirm = MsgBox("Are you sure you want to delete " _
            & fleLoop.Name, 4)
        If intConfirm = vbYes Then
            fleLoop.Delete
        End If
    End If
Next fleLoop

If blnMatch <> True Then
    MsgBox "There was no First Letter exception: " _
        & strException
End If
```



FirstLineIndent Property

-

Returns or sets the value (in points) for a first line or hanging indent. Use a positive value to set a first-line indent, and use a negative value to set a hanging indent. Read/write **Single**.

Example

This example sets a first-line indent of 1 inch for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).FirstLineIndent = _  
    InchesToPoints(1)
```

This example sets a hanging indent of 0.5 inch for the second paragraph in the active document. The **InchesToPoints** method is used to convert inches to points.

```
ActiveDocument.Paragraphs(2).FirstLineIndent = _  
    InchesToPoints(-0.5)
```



FirstPageTray Property

Returns or sets the paper tray to use for the first page of a document or section. Read/write [WdPaperTray](#).

WdPaperTray can be one of these WdPaperTray constants.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormatBin

wdPrinterTractorFeed

wdPrinterUpperBin

expression.**FirstPageTray**

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

Example

This example sets the tray to use for printing the first page of each section in the active document.

```
ActiveDocument.PageSetup.FirstPageTray = wdPrinterLowerBin
```

This example sets the tray to use for printing the first page of each section in the selection.

```
Selection.PageSetup.FirstPageTray = wdPrinterUpperBin
```



FirstRecord Property

-

Returns or sets the number of the first data record to be merged in a mail merge operation. Read/write **Long**.

Example

This example merges the main document with data records 1 through 3 and sends the merge documents to the printer.

```
With ActiveDocument.MailMerge
    .DataSource.FirstRecord = 1
    .DataSource.LastRecord = 3
    .Destination = wdSendToPrinter
    .Execute
End With
```



FitText Property

-

True if Microsoft Word visually reduces the size of text typed into a cell so that it fits within the column width. Read/write **Boolean**.

Remarks

If the **FitText** property is set to **True**, the font size of the text is not changed, but the visual width of the characters is adjusted to fit all the typed text into the cell.

Example

This example sets the first cell in the selection to automatically fit typed text within its width.

```
Selection.Cells(1).FitText = True
```



FitTextWidth Property

Returns or sets the width (in the current measurement units) in which Microsoft Word fits the text in the current selection or range. Read/write **Single**.

expression.**FitTextWidth**

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

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example fits the current selection into a space five centimeters wide.

```
Selection.FitTextWidth = CentimetersToPoints(5)
```



Flags Property

Returns or sets properties of the selection. Read/write [WdSelectionFlags](#).

WdSelectionFlags can be one of these WdSelectionFlags constants.

wdSelActive

wdSelOvertyp

wdSelStartActive

wdSelAtEOL

wdSelReplace

The return value of the **Flags** property is the sum of the **WdSelectionFlags** constants that apply to the selection.

Note: Setting the Flags property to wdSelAtEOL will make the end of the selection active.

expression.**Flags**

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

Example

This example selects the first word in the active document. The first message box displays "False" because the end of the selection is active. The **Flags** property makes the beginning of the selection active., and the second message box displays "True."

```
ActiveDocument.Words(1).Select  
MsgBox Selection.StartIsActive  
Selection.Flags = wdSelStartActive  
MsgBox Selection.StartIsActive
```

This example turns on overtype mode for the selection.

```
Selection.Flags = wdSelStartActive
```



↳ [Show All](#)

FlowDirection Property

Returns or sets the direction in which text flows from one text column to the next. Read/write [WdFlowDirection](#).

WdFlowDirection can be one of these WdFlowDirection constants.

wdFlowLtr Text in columns flows from left to right.

wdFlowRtl Text in columns flows from right to left.

expression.**FlowDirection**

expression Required. An expression that returns a [TextColumns](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the flow direction so that text flows through the specified columns from right to left.

```
ActiveDocument.PageSetup.TextColumns.FlowDirection = _  
    wdFlowRtl
```



FocusInMailHeader Property

-
True if the insertion point is in an e-mail header field (the **To:** field, for example). Read-only **Boolean**.

Example

This example displays a message in the status bar if the insertion point is in an e-mail header field.

```
If Application.FocusInMailHeader = True Then  
    StatusBar = "Selection is in message header"  
End If
```



↳ [Show All](#)

FolderSuffix Property

Returns the folder suffix that Microsoft Word uses when you save a document as a Web page, use long file names, and choose to save supporting files in a separate folder (that is, if the [UseLongFileNames](#) and [OrganizeInFolder](#) properties are set to **True**). Read-only **String**.

expression.**FolderSuffix**

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

Remarks

Newly created documents use the suffix returned by the **FolderSuffix** property of the **DefaultWebOptions** object. The value of the **FolderSuffix** property of the **WebOptions** object may differ from that of the **DefaultWebOptions** object if the document was previously edited in a different language version of Microsoft Word. You can use the [UseDefaultFolderSuffix](#) method to change the suffix to the language you are currently using in Microsoft Office.

By default, the name of the supporting folder is the name of the Web page plus an underscore (_), a period (.), or a hyphen (-) and the word "files" (appearing in the language of the version of Word in which the file was saved as a Web page). For example, suppose that you use the Dutch language version of Word to save a file called "Page1" as a Web page. The default name of the supporting folder is Page1_bestanden.

The following table lists each language version of Office and gives its corresponding [LanguageID](#) property value and folder suffix. For the languages that are not listed in the table, the suffix ".files" is used.

LanguageID property values

Language	LanguageID	Folder suffix
Arabic	1025	.files
Basque	1069	_fitxategiak
Brazilian	1046	_arquivos
Bulgarian	1026	.files
Catalan	1027	_fitxers
Chinese - Simplified	2052	.files
Chinese - Traditional	1028	.files
Croatian	1050	_datoteke
Czech	1029	_soubory
Danish	1030	-filer
Dutch	1043	_bestanden
English	1033	_files
Estonian	1061	_failid

Finnish	1035	_tiedostot
French	1036	_fichiers
German	1031	-Dateien
Greek	1032	.files
Hebrew	1037	.files
Hungarian	1038	_elemei
Italian	1040	_file
Japanese	1041	.files
Korean	1042	.files
Latvian	1062	_fails
Lithuanian	1063	_bylos
Norwegian	1044	-filer
Polish	1045	_pliki
Portuguese	2070	_ficheiros
Romanian	1048	.files
Russian	1049	.files
Serbian (Cyrillic)	3098	.files
Serbian (Latin)	2074	_fajlovi
Slovakian	1051	.files
Slovenian	1060	_datoteke
Spanish	3082	_archivos
Swedish	1053	-filer
Thai	1054	.files
Turkish	1055	_dosyalar
Ukranian	1058	.files
Vietnamese	1066	.files

Example

This example places the folder suffix used by the active document in a string variable.

```
strFolderSuffix = ActiveDocument.WebOptions.FolderSuffix
```



Font Property

-

Returns or sets a [Font](#) object that represents the character formatting of the specified object. To set this property, specify an expression that returns a **Font** object. Read/write **Font**.

Example

This example removes bold formatting from the Heading 1 style in the active document.

```
ActiveDocument.Styles(wdStyleHeading1).Font.Bold = False
```

This example toggles the font of the second paragraph in the active document between Arial and Times New Roman.

```
Set myRange = ActiveDocument.Paragraphs(2).Range
If myRange.Font.Name = "Times New Roman" Then
    myRange.Font.Name = "Arial"
Else
    myRange.Font.Name = "Times New Roman"
End If
```

This example displays the font of the selected text.

```
MsgBox Selection.Font.Name
```

This example applies the character formatting of the selected text to the first paragraph in the active document.

```
Set myFont = Selection.Font.Duplicate
ActiveDocument.Paragraphs(1).Range.Font = myFont
```

This example finds the next range of text that's formatted with the Times New Roman font.

```
With Selection.Find
    .ClearFormatting
    .Font.Name = "Times New Roman"
    .Execute FindText:="", ReplaceWith:"", Format:=True, _
        Forward:=True
End With
```



FontBold Property

-
Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue The font in the specified WordArt is bold.

Example

This example sets the font to bold for the third shape on the active document if the shape is WordArt.

```
Dim docActive As Document  
  
Set docActive = ActiveDocument  
  
With docActive.Shapes(3)  
    If .Type = msoTextEffect Then  
        .TextEffect.FontBold = msoTrue  
    End If  
End With
```



↳ [Show All](#)

FontItalic Property

-
Italicizes WordArt text. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**FontItalic**

expression Required. An expression that returns a [TextEffectFormat](#) object.

Example

This example sets the font to italic for the shape named "WordArt 4" in the active document.

```
Sub ItalicizeWordArt()  
    ActiveDocument.Shapes("WordArt 4") _  
        .TextEffect.FontItalic = msoTrue  
End Sub
```



FontName Property

-

Returns or sets the name of the font for the dropped capital letter. Read/write **String**.

Example

This example sets Arial as the font for the dropped capital letter for the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).DropCap
    .FontName = "Arial"
    .Position = wdDropNormal
    .LinesToDrop = 3
    .DistanceFromText = InchesToPoints(0.1)
End With
```



FontNames Property

-

Returns a [FontNames](#) object that includes the names of all the available fonts.
Read-only.

Example

This example displays the font names in the **FontNames** collection.

```
Dim strFont As String
Dim intResponse As Integer

For Each strFont In FontNames
    intResponse = MsgBox(Prompt:=strFont, Buttons:=vbOKCancel)
    If intResponse = vbCancel Then Exit For
Next strFont
```



Fonts Property

Returns the [WebPageFonts](#) collection representing the set of fonts Microsoft Word uses when you open a Web page in Word and either there is no font information specified in the Web page, or the current default font can't display the character set in the Web page.

expression.**Fonts**

expression Required. An expression that returns a [DefaultWebObtions](#) object.

Example

This example sets the default fixed-width font for the English/Western European/Other Latin Script character set to Courier New, 14 points.

```
With Application.DefaultWebOptions _  
    .Fonts(msoCharacterSetEnglishWesternEuropeanOtherLatinScript  
    .FixedWidthFont = "Courier New"  
    .FixedWidthFontSize = 14  
End With
```



FontSize Property

-

Returns or sets the font size for the specified WordArt, in points. Read/write **Single**.

Example

This example sets the font size to 16 points for the shape named "WordArt 2" in the active document.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
docActive.Shapes("WordArt 2").TextEffect.FontSize = 16
```



FooterDistance Property

-

Returns or sets the distance (in points) between the footer and the bottom of the page. Read/write **Single**.

Example

This example sets the distance between the footer and the bottom of the page to 0.5 inch. The **InchesToPoints** method is used to convert inches to points.

```
ActiveDocument.PageSetup.FooterDistance = InchesToPoints(0.5)
```

This example sets the distance between the footer and the bottom of the page for all the sections in the selection to 1 inch.

```
Selection.Range.PageSetup.FooterDistance = 72
```



Footers Property

Returns a [HeadersFooters](#) collection that represents the footers in the specified section. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a right-aligned page number to the primary footer in the first section in the active document.

```
With ActiveDocument.Sections(1).Footers(wdHeaderFooterPrimary)  
    .PageNumbers.Add PageNumberAlignment:=wdAlignPageNumberRight  
End With
```



FootnoteOptions Property

Returns [FootnoteOptions](#) object that represents the footnotes in a selection or range.

expression.**FootnoteOptions**

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

Example

This example sets the numbering rule in section two to restart at the beginning of the new section.

```
Sub SetFootnoteOptionsRange()  
    ActiveDocument.Sections(2).Range.FootnoteOptions _  
        .NumberingRule = wdRestartSection  
End Sub
```



Footnotes Property

-
Returns a [Footnotes](#) collection that represents all the footnotes in a range, selection, or document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example changes the footnote reference marks for the footnotes in the active document to lowercase letters, starting with the letter "c".

```
With ActiveDocument.Footnotes  
    .StartingNumber = 3  
    .NumberStyle = wdNoteNumberStyleLowercaseLetter  
End With
```

This example inserts an automatically numbered footnote at the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart  
ActiveDocument.Footnotes.Add Range:=Selection.Range, _  
    Text:="(Lone Creek Press, 1995)"
```



ForeColor Property

-

Returns or sets a [ColorFormat](#) object that represents the foreground color for the fill, line, or shadow. Read/write.

Example

This example adds a rectangle to the active document and then sets the foreground color, background color, and gradient for the rectangle's fill.

```
Dim docActive As Document

Set docActive = ActiveDocument

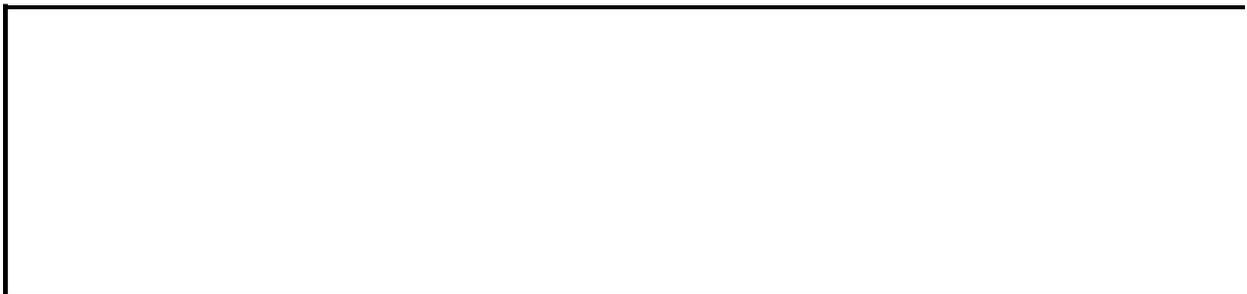
With docActive.Shapes.AddShape(msoShapeRectangle, _
    90, 90, 90, 50).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(170, 170, 170)
    .TwoColorGradient msoGradientHorizontal, 1
End With
```

This example adds a patterned line to the active document.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes.AddLine(10, 100, 250, 0).Line
    .Weight = 6
    .ForeColor.RGB = RGB(0, 0, 255)
    .BackColor.RGB = RGB(128, 0, 0)
    .Pattern = msoPatternDarkDownwardDiagonal
End With
```



↳ [Show All](#)

ForegroundColorProperty

Returns or sets the 24-bit color that's applied to the foreground of the [Shading](#) object. This color is applied to the dots and lines in the shading pattern. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua
wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender

wdColorLightGreen
wdColorLightTurquoise
wdColorLime
wdColorOrange
wdColorPink

expression.**ForegroundColor**

expression Required. An expression that returns a [Shading](#) object.

Example

This example applies shading with teal dots on a dark red background to the selection.

```
With Selection.Shading
    .Texture = wdTexture30Percent
    .ForegroundColor = wdColorTeal
    .BackgroundPatternColor = wdColorDarkRed
End With
```



ForegroundPatternColorIndex Property

Returns or sets the color that's applied to the foreground of the **Shading** object. This color is applied to the dots and lines in the shading pattern. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**ForegroundPatternColorIndex**

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

Example

This example applies shading with different foreground and background colors to the selection.

```
With Selection.Shading
    .Texture = wdTexture30Percent
    .ForegroundColorIndex = wdBlue
    .BackgroundPatternColorIndex = wdYellow
End With
```



↳ [Show All](#)

Format Property

▶ [Format property as it applies to the **Find** object.](#)

True if formatting is included in the find operation. Read/write **Boolean**.

expression.**Format**

expression Required. An expression that returns a [Find](#) object.

▶ [Format property as it applies to the **Indexes** object.](#)

Returns or sets the formatting for the indexes in the specified document. Read/write [WdIndexFormat](#).

WdIndexFormat can be one of these WdIndexFormat constants.

wdIndexBulleted

wdIndexFancy

wdIndexModern

wdIndexTemplate

wdIndexClassic

wdIndexFormal

wdIndexSimple

expression.**Format**

expression Required. An expression that returns an [Indexes](#) object.

▶ [Format property as it applies to the **Paragraph** and **Paragraphs** objects.](#)

Returns or sets a [ParagraphFormat](#) object that represents the formatting of the specified paragraph or paragraphs.

expression.**Format**

expression Required. An expression that returns one of the above objects.

▶ [Format property as it applies to the **TablesOfAuthorities** object.](#)

Returns or sets the formatting for the tables of authorities in the specified document. Read/write **WdToaFormat**.

WdToaFormat can be one of these WdToaFormat constants.

wdTOAClassic

wdTOAFormal

wdTOATemplate

wdTOADistinctive

wdTOASimple

expression.**Format**

expression Required. An expression that returns a [TablesOfAuthorities](#) object.

▶ [Format property as it applies to the **TablesOfContents** object.](#)

Returns or sets the formatting for the tables of contents in the specified document. Read/write **WdTocFormat**.

WdTocFormat can be one of these WdTocFormat constants.

wdTOCDistinctive

wdTOCFormal

wdTOCSimple

wdTOCClassic

wdTOCFancy

wdTOCModern

wdTOCTemplate

expression.**Format**

expression Required. An expression that returns a [TablesOfContents](#) object.

▶ [Format property as it applies to the **TablesOfFigures** object.](#)

Returns or sets the formatting for the tables of figures in the specified document. Read/write [WdTofFormat](#).

WdTofFormat can be one of these WdTofFormat constants.

wdTOFCentered

wdTOFDistinctive

wdTOFSimple

wdTOFClassic

wdTOFFormal

wdTOFTemplate

expression.**Format**

expression Required. An expression that returns a [TablesOfFigures](#) object.

▶ [Format property as it applies to the TextInput object.](#)

Returns the text formatting for the specified text box. Read-only **String**.

expression.**Format**

expression Required. An expression that returns a [TextInput](#) object.

Example

▶ [As it applies to the **Find** object.](#)

This example removes all bold formatting in the active document.

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Font.Bold = True
    .Format = True
    .Replacement.ClearFormatting
    .Replacement.Font.Bold = False
    .Execute Forward:=True, Replace:=wdReplaceAll, _
        FindText:="", ReplaceWith:=""
End With
```

▶ [As it applies to the **Paragraph** object.](#)

This example returns the formatting of the first paragraph in the active document and then applies the formatting to the selection.

```
Set paraFormat = ActiveDocument.Paragraphs(1).Format.Duplicate
Selection.Paragraphs.Format = paraFormat
```

▶ [As it applies to the **Paragraphs** object.](#)

The following example left-aligns all the paragraphs in the active document.

```
ActiveDocument.Paragraphs.Format.Alignment = wdAlignParagraphLeft
```

▶ [As it applies to the **TablesOfContents** object.](#)

This example applies Classic formatting to the tables of contents in Report.doc.

```
Documents("Report.doc").TablesOfContents.Format = wdTOCClassic
```

▶ [As it applies to the **TextInput** object.](#)

This example displays the text formatting in the first field of the active document.

```
If ActiveDocument.FormFields(1).Type = wdFieldFormTextInput Then
    MsgBox ActiveDocument.FormFields(1).TextInput.Format
Else
    MsgBox "First field is not a text form field"
End If
```



FormatDescription Property

Returns a **String** representing a description of tracked formatting changes in a revision. Read-only.

expression.**FormatDescription**

expression Required. An expression that returns a [Revision](#) object.

Example

This example displays a description for each of the formatting changes made in a document with tracked changes.

```
Sub FmtChanges()  
    Dim revFmtRev As Revision  
  
    For Each revFmtRev In ActiveDocument.Revisions  
        If revFmtRev.FormatDescription <> "" Then  
            MsgBox "Format changes made : " & revFmtRev.FormatDescri  
        End If  
    Next  
End Sub
```



FormatName Property

-

Returns the name of the specified file converter. The format names appear in the **Save as type** box in the **Save As** dialog box (**File** menu). Read-only **String**.

Example

This example displays the format name of the first converter in the **FileConverters** collection.

```
MsgBox FileConverters(1).FormatName
```

This example uses the AvailableConv() array to store the names of all the available file converters.

```
Dim intTemp As Integer
Dim fcLoop As FileConverter
Dim AvailableConv As Variant

ReDim AvailableConv(FileConverters.Count - 1)

intTemp = 0

For Each fcLoop In FileConverters
    AvailableConv(intTemp) = fcLoop.FormatName
    intTemp = intTemp + 1
Next fcLoop
```



FormatScanning Property

True for Microsoft Word to keep track of all formatting in a document.
Read/write **Boolean**.

expression.**FormatScanning**

expression Required. An expression that returns an [Options](#) object.

Remarks

Enabling the **FormatScanning** property allows you to identify all unique formatting in your document, so you can easily apply the same formatting to new text and quickly replace or modify all instances of a given formatting within a document.

Example

This example enables Word to keep track of formatting in documents but disables displaying a squiggly underline beneath text formatted similarly to other formatting that is used more frequently in a document.

```
Sub ShowFormatErrors()  
  With Options  
    .FormatScanning = True  
    .ShowFormatError = False 'Disables displaying squiggly unde  
  End With  
End Sub
```



FormattedText Property

-

Returns or sets a [Range](#) object that includes the formatted text in the specified range or selection. Read/write.

Remarks

This property returns a **Range** object with the character formatting and text from the specified range or selection. Paragraph formatting is included in the **Range** object if there's a paragraph mark in the range or selection.

When you set this property, the text in the range is replaced with formatted text. If you don't want to replace the existing text, use the **Collapse** method before using this property (see the first example).

Example

This example copies the first paragraph in the document, including its formatting, and inserts the formatted text at the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart  
Selection.FormattedText = ActiveDocument.Paragraphs(1).Range
```

This example copies the text and formatting from the selection into a new document.

```
Set myRange = Selection.FormattedText  
Documents.Add.Content.FormattedText = myRange
```



FormattingShowClear Property

True for Microsoft Word to show clear formatting in the **Styles and Formatting** task pane. Read/write **Boolean**.

expression.**FormattingShowClear**

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

Example

This example disables display of the **Clear Formatting** button in the list of styles.

```
Sub ShowClearFormatting()  
  With ActiveDocument  
    .FormattingShowClear = False  
    .FormattingShowFilter = wdShowFilterFormattingInUse  
    .FormattingShowFont = True  
    .FormattingShowNumbering = True  
    .FormattingShowParagraph = True  
  End With  
End Sub
```



↳ [Show All](#)

FormattingShowFilter Property

Sets or returns a [WdShowFilter](#) constant that represents the styles and formatting displayed in the **Styles and Formatting** task pane. Read/write **Boolean**.

WdShowFilter can be one of these WdShowFilter constants.

wdShowFilterFormattingAvailable

wdShowFilterFormattingInUse

wdShowFilterStylesAll

wdShowFilterStylesAvailable

wdShowFilterStylesInUse

expression.**FormattingShowFilter**

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

Example

This example filters formatting to show in the **Styles and Formatting** task pane only the formatting in use in the active document.

```
Sub ShowClearFormatting()  
  With ActiveDocument  
    .FormattingShowClear = False  
    .FormattingShowFilter = wdShowFilterFormattingInUse  
    .FormattingShowFont = True  
    .FormattingShowNumbering = True  
    .FormattingShowParagraph = True  
  End With  
End Sub
```



FormattingShowFont Property

True for Microsoft Word to display font formatting in the **Styles and Formatting** task pane. Read/write **Boolean**.

expression.**FormattingShowFont**

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

Example

This example enables display of font formatting in the **Styles and Formatting** task pane.

```
Sub ShowClearFormatting()  
  With ActiveDocument  
    .FormattingShowClear = False  
    .FormattingShowFilter = wdShowFilterFormattingInUse  
    .FormattingShowFont = True  
    .FormattingShowNumbering = True  
    .FormattingShowParagraph = True  
  End With  
End Sub
```



FormattingShowNumbering Property

True for Microsoft Word to display number formatting in the **Styles and Formatting** task pane. Read/write **Boolean**.

expression.**FormattingShowNumbering**

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

Example

This example enables displaying number formatting in the **Styles and Formatting** pane.

```
Sub ShowClearFormatting()  
  With ActiveDocument  
    .FormattingShowClear = False  
    .FormattingShowFilter = wdShowFilterFormattingInUse  
    .FormattingShowFont = True  
    .FormattingShowNumbering = True  
    .FormattingShowParagraph = True  
  End With  
End Sub
```



FormattingShowParagraph Property

True for Microsoft Word to display paragraph formatting in the **Styles and Formatting** task pane. Read/write **Boolean**.

expression.**FormattingShowParagraph**

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

Example

This example enables displaying paragraph formatting in the **Styles and Formatting** task pane.

```
Sub ShowClearFormatting()  
  With ActiveDocument  
    .FormattingShowClear = False  
    .FormattingShowFilter = wdShowFilterFormattingInUse  
    .FormattingShowFont = True  
    .FormattingShowNumbering = True  
    .FormattingShowParagraph = True  
  End With  
End Sub
```



FormFields Property

Returns a [FormFields](#) collection that represents all the form fields in the document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the content of the form field named "Text1" to "Name."

```
ActiveDocument.FormFields("Text1").Result = "Name"
```

This example retrieves the type of the first form field in section two.

```
myType = ActiveDocument.Sections(2).Range.FormFields(1).Type  
Select Case myType  
    Case wdFieldFormTextInput  
        thetype = "TextBox"  
    Case wdFieldFormDropDown  
        thetype = "DropDown"  
    Case wdFieldFormCheckBox  
        thetype = "CheckBox"  
End Select
```

This example displays the name of the first form field in the selection.

```
If Selection.FormFields.Count > 0 Then  
    MsgBox Selection.FormFields(1).Name  
End If
```



FormsDesign Property

True if the specified document is in form design mode. Read-only **Boolean**.

Note This property always returns **False** if it's used in code run from Microsoft Word, but it returns the correct value if it is run through Automation. For example, if you run the example from Microsoft Excel, it will return **True** if you're in design mode.

Remarks

When Word is in form design mode, the **Control Toolbox** toolbar is displayed. You can use this toolbar to insert ActiveX controls such as command buttons, scroll bars, and option buttons. In form design mode, event procedures don't run, and when you click an embedded control, the control's sizing handles appear.

Example

This example displays a message box that indicates whether the active document is in form design mode. This example will always return **False**.

```
Msgbox ActiveDocument.FormsDesign
```



Forward Property

-

True if the find operation searches forward through the document. **False** if it searches backward through the document. Read/write **Boolean**.

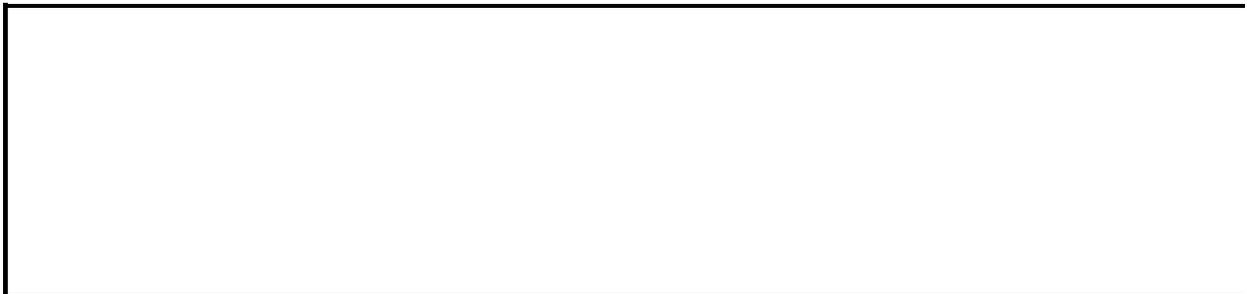
Example

This example replaces the next occurrence of the word "hi" in the selection with "hello."

```
With Selection.Find
    .Forward = True
    .Text = "hi"
    .ClearFormatting
    .Replacement.Text = "hello"
    .Execute Replace:=wdReplaceOne
End With
```

The following example searches backward through the document for the word "Microsoft." If the word is found, it's automatically selected.

```
Selection.Collapse Direction:=wdCollapseStart
With Selection.Find
    .Forward = False
    .ClearFormatting
    .MatchWholeWord = True
    .MatchCase = False
    .Wrap = wdFindContinue
    .Execute FindText:="Microsoft"
End With
```



Found Property

-
SynonymInfo object: **True** if the thesaurus finds synonyms, antonyms, related words, or related expressions for the word or phrase. Read-only **Boolean**.

Find object: **True** if the search produces a match. Read-only **Boolean**.

Example

This example checks to see whether the thesaurus contains any synonym suggestions for the word "authorize."

```
Dim siTemp As SynonymInfo

Set siTemp = SynonymInfo(Word:="authorize", _
    LanguageID:=wdEnglishUS)
If siTemp.Found = True Then
    MsgBox "The thesaurus has suggestions."
Else
    MsgBox "The thesaurus has no suggestions."
End If
```

This example checks to see whether the thesaurus contains any synonym suggestions for the selection. If it does, the example displays the **Thesaurus** dialog box with the synonyms listed.

```
Dim siTemp As SynonymInfo

Set siTemp = Selection.Range.SynonymInfo
If siTemp.Found = True Then
    Selection.Range.CheckSynonyms
Else
    MsgBox "The thesaurus has no suggestions."
End If
```

This example removes formatting from the find criteria before searching the selection. If the word "Hello" with bold formatting is found, the entire paragraph is selected and copied to the Clipboard.

```
With Selection.Find
    .ClearFormatting
    .Font.Bold = True
    .Execute FindText:="Hello", Format:=True, Forward:=True
    If .Found = True Then
        .Parent.Expand Unit:=wdParagraph
        .Parent.Copy
    End If
End With
```



Frame Property

-

Returns a [Frame](#) object that represents the frame formatting for the specified style or find-and-replace operation. Read-only.

Example

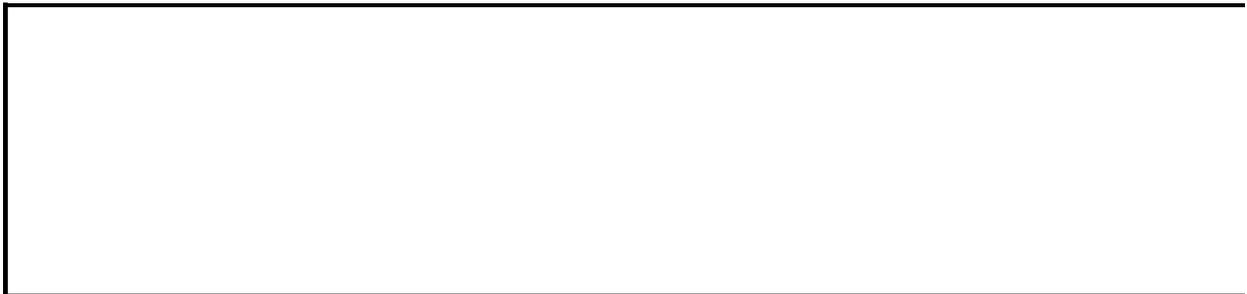
This example creates a style with frame formatting and then applies the style to the first paragraph in the selection.

```
Dim styleNew As Style

Set styleNew = ActiveDocument.Styles _
    .Add(Name:="frame", Type:=wdStyleTypeParagraph)
With styleNew.Frame
    .RelativeHorizontalPosition = _
        wdRelativeHorizontalPositionMargin
    .HeightRule = wdFrameAuto
    .WidthRule = wdFrameAuto
    .TextWrap = True
End With
Selection.Paragraphs(1).Range.Style = "frame"
```

This example finds the first frame with wrap around formatting. If such a frame is found, a message is displayed on the status bar.

```
With ActiveDocument.Content.Find
    .Text = ""
    .Frame.TextWrap = True
    .Execute Forward:=True, Wrap:=wdFindContinue, Format:=True
    If .Found = True Then StatusBar = "Frame was found"
    .Parent.Select
End With
```



FrameDefaultURL Property

-

Returns or sets the Web page or other document to be displayed in the specified frame when the frames page is opened. Read/write **String**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the specified frame to display a local file named "Order.htm".

```
With ActiveDocument.ActiveWindow.ActivePane.Frameset
    .FrameDefaultURL = "C:\Documents\Order.htm"
    .FrameLinkToFile = True
End With
```



FrameDisplayBorders Property

-
True if the frame borders on the specified frames page are displayed. Read/write **Boolean**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets Microsoft Word to display frame borders in the specified frames page.

```
ActiveDocument.ActiveWindow.ActivePane.FrameSet _  
    .FrameDisplayBorders = True
```



FrameLinkToFile Property

-

True if the Web page or other document specified by the [FrameDefaultURL](#) property is an external file to which Microsoft Word maintains only a link from the specified frame. Read/write **Boolean**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets Microsoft Word to maintain only a link from the specified frame to the document "Order.htm".

```
With ActiveDocument.ActiveWindow.ActivePane.Frameset
    .FrameDefaultURL = "C:\Documents\Order.htm"
    .FrameLinkToFile = True
End With
```



FrameName Property

-

Returns or sets the name of the specified frame on a frames page. Read/write **String**.

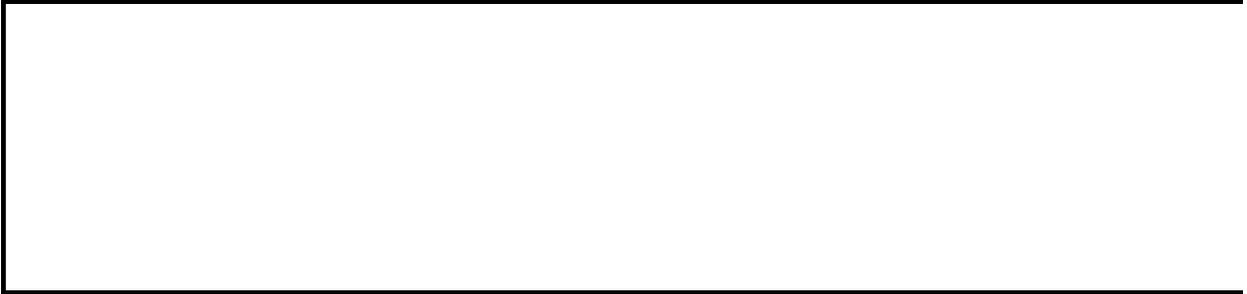
Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the name of the specified frame to "BottomFrame".

```
ActiveWindow.Document.Frameset _  
    .ChildFramesetItem(3).FrameName = "BottomFrame"
```



FrameResizable Property

-

True if the user can resize the specified frame when the frames page is viewed in a Web browser. Read/write **Boolean**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the specified frame to be resizable when viewed in a Web browser.

```
With ActiveDocument.ActiveWindow.ActivePane.Frameset
    .FrameDefaultURL = "C:\Documents\Order.htm"
    .FrameResizable = True
End With
```



Frames Property

-
Returns a [Frames](#) collection that represents all the frames in a document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example causes text to wrap around frames in the first section in the active document.

```
For Each aFrame In ActiveDocument.Sections(1).Range.Frames  
    aFrame.TextWrap = True  
Next aFrame
```

This example adds a frame around the selection and returns a frame object to the myFrame variable.

```
Set myFrame = ActiveDocument.Frames.Add(Range:=Selection.Range)
```



FrameScrollbarType Property

Returns or sets when scroll bars are available for the specified frame when viewing its frames page in a Web browser. Read/write [WdScrollbarType](#).

WdScrollbarType can be one of these WdScrollbarType constants.

wdScrollbarTypeNo Scroll bars are never available for the specified frame.

wdScrollbarTypeAuto Scroll bars are available for the specified frame only if the contents are too large to fit in the allotted space.

wdScrollbarTypeYes Scroll bars are always available for the specified frame.

expression.**FrameScrollbarType**

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

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example makes scroll bars always available for the specified frame, regardless of whether the contents of the frame require scrolling.

```
With ActiveDocument.ActiveWindow.ActivePane.FrameSet
    .FrameDefaultURL = "C:\Documents\Order.htm"
    .FrameScrollBarType = wdScrollBarTypeYes
End With
```



Frameset Property

-
Returns a [Frameset](#) object that represents an entire frames page or a single frame on a frames page. Read-only.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

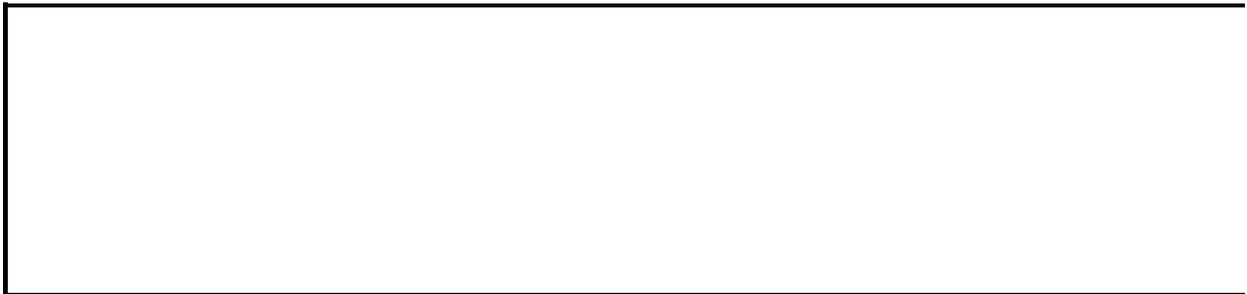
Example

This example sets the color of frame borders in the specified frames page to tan.

```
With ActiveWindow.Document.Frameset  
    .FramesetBorderColor = wdColorTan  
    .FramesetBorderWidth = 6  
End With
```

This example adds a new frame to the immediate right of the specified frame.

```
ActiveDocument.ActiveWindow.ActivePane.Frameset _  
    .AddNewFrame wdFramesetNewRight
```



↳ [Show All](#)

FramesetBorderColor Property

Returns or sets the color of the frame borders on the specified frames page. Can be any of the [WdColor](#) constants or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**FramesetBorderColor**

expression Required. An expression that returns a [Frameset](#) object.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the color of frame borders in the specified frames page to tan.

```
With ActiveWindow.Document.Frameset  
    .FramesetBorderColor = wdColorTan  
    .FramesetBorderWidth = 6  
End With
```



FramesetBorderWidth Property

-

Returns or sets the width (in points) of the borders surrounding the frames on the specified frames page. Read/write **Single**.

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example sets the width of frame borders in the specified frames page to 6 points.

```
With ActiveWindow.Document.Frameset  
    .FramesetBorderColor = wdColorTan  
    .FramesetBorderWidth = 6  
End With
```



FreeDiskSpace Property

Returns the available disk space for the current drive, in bytes. Use the **ChDrive** statement to change the current drive. Read-only **Long**.

Note There are 1024 bytes in a kilobyte and 1,048,576 bytes in a megabyte. The maximum return value for the **FreeDiskSpace** property is 2,147,483,647. Therefore, even if you have four gigabytes of free disk space, it returns 2147483647.

Example

This example checks the amount of free disk space. If there's less than 10 megabytes of space available, a message is displayed.

```
If (System.FreeDiskSpace \ 1048576) < 10 Then _  
    MsgBox "Low disk space"
```



FullName Property

-
Specifies the name of a document, template, or cascading style sheet, including the drive or Web path. Read-only **String**.

expression.**FullName**

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

Remarks

Using this property is equivalent to using the [Path](#), [PathSeparator](#), and [Name](#) properties in sequence.

Example

This example displays the path and file name of the active document.

```
Sub DocName()  
    MsgBox ActiveDocument.FullName  
End Sub
```

This example displays the path and file name of the template attached to the active document.

```
Sub TemplateName()  
    MsgBox ActiveDocument.AttachedTemplate.FullName  
End Sub
```

This example displays the path and file name of the style sheet attached to the active document.

```
Sub CSSName()  
    MsgBox ActiveDocument.StyleSheets(1).FullName  
End Sub
```



FullScreen Property

-

True if the window is in full-screen view. Read/write **Boolean**.

Example

This example switches the active window to full-screen view.

```
ActiveDocument.ActiveWindow.View.FullScreen = True
```

This example activates the window for Sales.doc and switches out of full-screen view.

```
With Windows("Sales.doc")  
    .Activate  
    .View.FullScreen = False  
End With
```



Gap Property

-

Returns or sets the horizontal distance (in points) between the end of the callout line and the text bounding box. Read/write **Single**.

Example

This example sets the distance between the callout line and the text bounding box to 3 points for the first shape on the active document. For the example to work, the first shape must be a callout.

```
Dim docActive As Document
```

```
Set docActive = ActiveDocument
```

```
docActive.Shapes(1).Callout.Gap = 3
```



GradientColorType Property

Returns the gradient color type for the specified fill. Read-only
[MsoGradientColorType](#).

MsoGradientColorType can be one of these MsoGradientColorType constants.

msoGradientColorMixed

msoGradientOneColor

msoGradientPresetColors

msoGradientTwoColors

expression.**GradientColorType**

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

This property is read-only. Use the [OneColorGradient](#), [PresetGradient](#), or [TwoColorGradient](#) method to set the gradient type for the fill.

Example

This example changes the fill for all shapes in the active document that have a two-color gradient fill to a preset gradient fill.

```
Dim docActive As Document
Dim shapeLoop As Shape

Set docActive = ActiveDocument

For Each shapeLoop In docActive.Shapes
    With shapeLoop.Fill
        If .GradientColorType = msoGradientTwoColors Then
            .PresetGradient msoGradientHorizontal, 1, _
                msoGradientBrass
        End If
    End With
Next
```



GradientDegree Property

Returns a value that indicates 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 **Single**.

This property is read-only. Use the [OneColorGradient](#) method to set the gradient degree for the fill.

Example

This example adds a rectangle to the active document 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 fails.

```
Dim docActive As Document
Dim sngGradient As Single

Set docActive = ActiveDocument

With docActive.Shapes
    sngGradient = .Item("Rectangle 2").Fill.GradientDegree
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        .ForeColor.RGB = RGB(0, 128, 128)
        .OneColorGradient msoGradientHorizontal, 1, sngGradient
    End With
End With
```



GradientStyle Property

Returns the gradient style for the specified fill. Read-only [MsoGradientStyle](#).

MsoGradientStyle can be one of these MsoGradientStyle constants.

msoGradientDiagonalDown

msoGradientDiagonalUp

msoGradientFromCenter

msoGradientFromCorner

msoGradientFromTitle Only used with Microsoft PowerPoint.

msoGradientHorizontal

msoGradientMixed

msoGradientVertical

expression.**GradientStyle**

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

This property is read-only. Use the [OneColorGradient](#) or [TwoColorGradient](#) method to set the gradient style for the fill.

Note 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 document 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 docActive As Document
Dim lngGradient As Long
```

```
Set docActive = ActiveDocument
```

```
With docActive.Shapes
    lngGradient = .Item("rect1").Fill.GradientStyle
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient lngGradient, 1, 1
    End With
End With
```



GradientVariant Property

Returns the gradient variant for the specified fill as an integer value from 1 to 4 for most gradient fills. If the gradient style is **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 **Long**.

This property is read-only. Use the [OneColorGradient](#) or [TwoColorGradient](#) method to set the gradient variant for the fill.

Example

This example adds a rectangle to the active document 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 lngGradient As Long
```

```
With ActiveDocument.Shapes
```

```
    lngGradient = .Item("rect1").Fill.GradientVariant
```

```
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
```

```
        .ForeColor.RGB = RGB(128, 0, 0)
```

```
        .OneColorGradient msoGradientHorizontal, _
```

```
            lngGradient, 1
```

```
    End With
```

```
End With
```



GrammarChecked Property

-

True if a grammar check has been run on the specified range or document. **False** if some of the specified range or document hasn't been checked for grammar.
Read/write **Boolean**.

Remarks

To recheck the grammar in a range or document, set the **GrammarChecked** property to **False**.

Example

This example determines whether grammar has been checked in the active document. If it has, the word count is displayed. If grammar hasn't been checked, a spelling and grammar check is started.

```
Set myStat = ActiveDocument.ReadabilityStatistics
passGram = ActiveDocument.GrammarChecked
If passGram = True Then
    MsgBox myStat(1).Name & " - " & myStat(1).Value
Else
    ActiveDocument.CheckGrammar
End If
```

This example sets the **GrammarChecked** property to **False** for the active document, and then it runs a grammar check again.

```
ActiveDocument.GrammarChecked = False
ActiveDocument.CheckGrammar
```



GrammaticalErrors Property

Returns a [ProofreadingErrors](#) collection that represents the sentences that failed the grammar check on the specified document or range. There can be more than one error per sentence. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Remarks

If there are no grammatical errors, the **Count** property for the **ProofreadingErrors** object returned by the **GrammaticalErrors** property returns 0 (zero).

Example

This example checks the third paragraph in the active document for grammatical errors and displays each sentence that contains one or more errors.

```
Set myErrors = ActiveDocument.Paragraphs(3).Range.GrammaticalErrors
For Each myerr In myErrors
    MsgBox myerr.Text
Next myerr
```

This example checks the active document for grammatical errors. If any errors are found, a new spelling and grammar check is started.

```
If ActiveDocument.GrammaticalErrors.Count = 0 Then
    MsgBox "There are no grammatical errors."
Else
    ActiveDocument.CheckGrammar
End If
```



GridDistanceHorizontal Property

-

Document object: Returns or sets the amount of horizontal space between the invisible gridlines that Microsoft Word uses when you draw, move, and resize AutoShapes or East Asian characters in the specified document. Read/write **Single**.

Options object: Returns or sets the amount of horizontal space between the invisible gridlines that Word uses when you draw, move, and resize AutoShapes or East Asian characters in new documents. Read/write **Single**.

Example

This example sets the horizontal and vertical distance between gridlines and then enables the **Snap objects to grid** feature for the current document.

```
With ActiveDocument
    .GridDistanceHorizontal = 9
    .GridDistanceVertical = 9
    .SnapToGrid = True
End With
```

This example sets the horizontal and vertical distance between gridlines and then enables the **Snap objects to grid** feature for a new document.

```
With Options
    .GridDistanceHorizontal = InchesToPoints(0.2)
    .GridDistanceVertical = InchesToPoints(0.2)
    .SnapToGrid = True
End With
Documents.Add
```



GridDistanceVertical Property

-

Document object: Returns or sets the amount of vertical space between the invisible gridlines that Microsoft Word uses when you draw, move, and resize AutoShapes or East Asian characters in the specified document. Read/write **Single**.

Options object: Returns or sets the amount of vertical space between the invisible gridlines that Word uses when you draw, move, and resize AutoShapes or East Asian characters in new documents. Read/write **Single**.

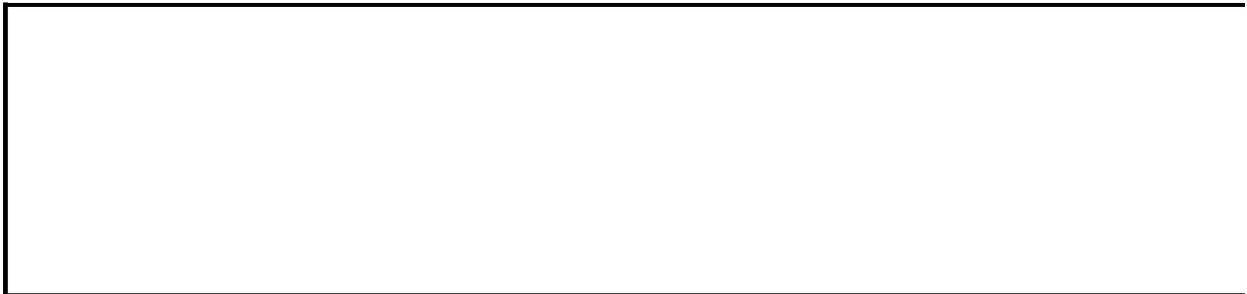
Example

This example sets the horizontal and vertical distance between gridlines and then enables the **Snap objects to grid** feature for the current document.

```
With ActiveDocument
    .GridDistanceHorizontal = 9
    .GridDistanceVertical = 9
    .SnapToGrid = True
End With
```

This example sets the horizontal and vertical distance between gridlines and then enables the **Snap objects to grid** feature for a new document.

```
With Options
    .GridDistanceHorizontal = InchesToPoints(0.2)
    .GridDistanceVertical = InchesToPoints(0.2)
    .SnapToGrid = True
End With
Documents.Add
```



GridOriginFromMargin Property

-
True if Microsoft Word starts the character grid from the upper-left corner of the page. Read/write **Boolean**.

Example

This example sets Microsoft Word to start the character grid for the active document from the upper-left corner of the page.

```
ActiveDocument.GridOriginFromMargin = True
```



GridOriginHorizontal Property

-

Document object: Returns or sets the point, relative to the left edge of the page, where you want the invisible grid for drawing, moving, and resizing AutoShapes or East Asian characters to begin in the specified document. Read/write **Single**.

Options object: Returns or sets the point, relative to the left edge of the page, where you want the invisible grid for drawing, moving, and resizing AutoShapes or East Asian characters to begin in new documents. Read/write **Single**.

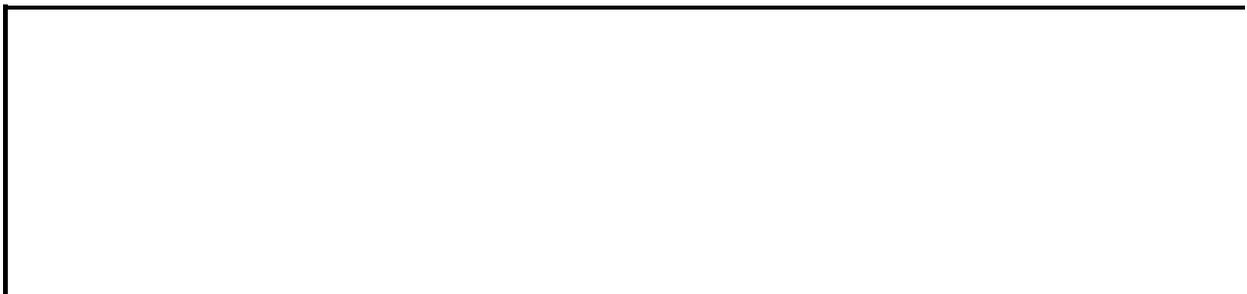
Example

This example sets the horizontal and vertical point of origin for the grid, sets the horizontal and vertical distance between gridlines, and then enables the **Snap to grid** feature for the current document.

```
With ActiveDocument
    .GridOriginHorizontal = 80
    .GridOriginVertical = 90
    .GridDistanceHorizontal = 9
    .GridDistanceVertical = 9
    .SnapToGrid = True
End With
```

This example sets the horizontal and vertical point of origin for the grid, sets the horizontal and vertical distance between gridlines, and then enables the **Snap objects to grid** feature for a new document.

```
With Options
    .GridOriginHorizontal = InchesToPoints(1)
    .GridOriginVertical = InchesToPoints(2)
    .GridDistanceHorizontal = InchesToPoints(0.1)
    .GridDistanceVertical = InchesToPoints(0.1)
    .SnapToGrid = True
End With
Documents.Add
```



GridOriginVertical Property

Document object: Returns or sets the point, relative to the top of the page, where you want the invisible grid for drawing, moving, and resizing AutoShapes or East Asian characters to begin in the specified document. Read/write **Single**.

Options object: Returns or sets the point, relative to the top of the page, where you want the invisible grid for drawing, moving, and resizing AutoShapes or East Asian characters to begin in new documents. Read/write **Single**.

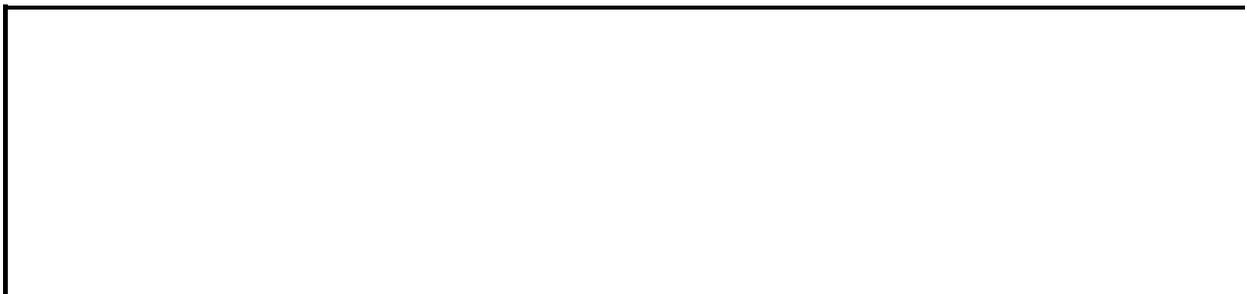
Example

This example sets the horizontal and vertical point of origin for the grid, sets the horizontal and vertical distance between gridlines, and then enables the **Snap objects to grid** feature for the current document.

```
With ActiveDocument
    .GridOriginHorizontal = 80
    .GridOriginVertical = 90
    .GridDistanceHorizontal = 9
    .GridDistanceVertical = 9
    .SnapToGrid = True
End With
```

This example sets the horizontal and vertical point of origin for the grid, sets the horizontal and vertical distance between gridlines, and then enables the **Snap objects to grid** feature for a new document.

```
With Options
    .GridOriginHorizontal = InchesToPoints(1)
    .GridOriginVertical = InchesToPoints(2)
    .GridDistanceHorizontal = InchesToPoints(0.2)
    .GridDistanceVertical = InchesToPoints(0.2)
    .SnapToGrid = True
End With
Documents.Add
```



GridSpaceBetweenHorizontalLines Property

Returns or sets the interval at which Microsoft Word displays horizontal character gridlines in print layout view. Read/write **Long**.

Example

This example sets Microsoft Word to display every fifth horizontal character gridline.

```
ActiveDocument.GridSpaceBetweenHorizontalLines = 5
```



GridSpaceBetweenVerticalLines Property

Returns or sets the interval at which Microsoft Word displays vertical character gridlines in print layout view. Read/write **Long**.

Example

This example sets Microsoft Word to display every other vertical character gridline.

```
ActiveDocument.GridSpaceBetweenVerticalLines = 2
```



GroupItems Property

-

Returns a [GroupShapes](#) object that represents the individual shapes in the specified group. Use the [Item](#) method of the **GroupShapes** object to return a single shape from the group. Applies to **Shape** or **ShapeRange** objects that represent grouped shapes. Read-only.

Example

This example adds three triangles to myDocument, groups them, sets a color for the entire group, and then changes the color for the second triangle only.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    .AddShape(msoShapeIsoscelesTriangle, _
        10, 10, 100, 100).Name = "shpOne"
    .AddShape(msoShapeIsoscelesTriangle, _
        150, 10, 100, 100).Name = "shpTwo"
    .AddShape(msoShapeIsoscelesTriangle, _
        300, 10, 100, 100).Name = "shpThree"
    With .Range(Array("shpOne", "shpTwo", "shpThree")).Group
        .Fill.PresetTextured msoTextureBlueTissuePaper
        .GroupItems(2).Fill.PresetTextured msoTextureGreenMarble
    End With
End With
```



Gutter Property

-

Returns or sets the amount (in points) of extra margin space added to each page in a document or section for binding. Read/write **Single**.

Remarks

If the **MirrorMargins** property is set to **True**, the **Gutter** property adds the extra space to the inside margins. Otherwise, the extra space is added to the left margin.

Example

This example adds 1 inch (72 points) to the inside margins of the active document.

```
With ActiveDocument.PageSetup  
    .MirrorMargins = True  
    .Gutter = 72  
End With
```



↳ [Show All](#)

GutterPos Property

Returns or sets on which side the gutter appears in a document. Read/write [WdGutterStyle](#).

WdGutterStyle can be one of these WdGutterStyle constants.

wdGutterPosLeft

wdGutterPosRight

wdGutterPosTop

expression.**GutterPos**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the gutter to appear on the right side of the document.

```
ActiveDocument.PageSetup.GutterPos = wdGutterPosRight
```



↳ [Show All](#)

GutterStyle Property

-

Returns or sets whether Microsoft Word uses gutters for the current document based on a right-to-left language or a left-to-right language. Read/write [WdGutterStyleOld](#).

WdGutterStyleOld can be one of these WdGutterStyleOld constants.

wdGutterStyleLatin

wdGutterStyleBidi

expression.**GutterStyle**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the current document to follow a gutter style for a right-to-left language document.

```
ActiveDocument.PageSetup.GutterStyle = wdGutterStyleBidi
```



HalfWidthPunctuationOnTopOfLine Property

True if Microsoft Word changes punctuation symbols at the beginning of a line to half-width characters for the specified paragraphs. This property returns **wdUndefined** if it's set to **True** for only some of the specified paragraphs.
Read/write **Long**.

Example

This example sets Microsoft Word to change punctuation symbols at the beginning of a line to half-width characters for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).HalfWidthPunctuationOnTopOfLine = True
```



HangingPunctuation Property

-

True if hanging punctuation is enabled for the specified paragraphs. This property returns **wdUndefined** if it's set to **True** for only some of the specified paragraphs. Read/write **Long**.

Example

This example enables hanging punctuation for the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).HangingPunctuation = True
```



HangulAndAlphabetAutoAdd Property

True if Microsoft Word automatically adds words to the list of Hangul and alphabet AutoCorrect exceptions on the **Korean** tab in the **AutoCorrect Exceptions** dialog box (on the **Tools** menu, click **AutoCorrect Options**, then click the **AutoCorrect** tab, and then click the **Exceptions** button). Word adds a word to this list if you delete and then retype a word that you didn't want Word to correct. Read/write **Boolean**.

expression.**HangulAndAlphabetAutoAdd**

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets Microsoft Word to automatically add words to the list of hangul and alphabet AutoCorrect exceptions on the **Korean** tab in the **AutoCorrect Exceptions** dialog box.

```
AutoCorrect.HangulAndAlphabetAutoAdd = True
```



HangulAndAlphabetExceptions Property

Returns a [HangulAndAlphabetExceptions](#) collection that represents the list of Hangul and alphabet AutoCorrect exceptions. This list corresponds to the list of Hangul and alphabet AutoCorrect exceptions on the **Korean** tab in the **AutoCorrect Exceptions** dialog box (on the **Tools** menu, click **AutoCorrect Options**, then click the **AutoCorrect** tab, and then click the **Exceptions** button).

expression.**HangulAndAlphabetExceptions**

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example prompts the user to delete or keep each hangul and alphabet AutoCorrect exception on the **Korean** tab in the **AutoCorrect Exceptions** dialog box.

```
For Each anEntry In _
    AutoCorrect.HangulAndAlphabetExceptions
    response = MsgBox("Delete entry: " _
        & anEntry.Name, vbYesNoCancel)
    If response = vbYes Then
        anEntry.Delete
    Else
        If response = vbCancel Then End
    End If
Next anEntry
```



HangulHanjaDictionaries Property

Returns a [HangulHanjaConversionDictionaries](#) collection that represents all the active custom conversion dictionaries. Active custom conversion dictionaries are marked with a check in the **Custom Dictionaries** dialog box (on the **Tools** menu, click **Options**, then click the **Spelling & Grammar** tab, and then click the **Custom Dictionaries** button).

expression.**HangulHanjaDictionaries**

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](#).

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example adds a new, blank custom dictionary to the collection. The path and file name of the new custom dictionary are then displayed in a message box.

```
Set myHome = _  
    HangulHanjaDictionaries.Add(Filename:="Home.hhd")  
Msgbox myHome.Path & Application.PathSeparator _  
    & myHome.Name
```

This example deactivates all custom dictionaries but does not delete the custom dictionary files.

```
HangulHanjaDictionaries.ClearAll
```

This example displays the name of each custom dictionary in the collection.

```
For Each di In HangulHanjaDictionaries  
    MsgBox di.Name  
Next di
```



HangulHanjaFastConversion Property

True if Microsoft Word automatically converts a word with only one suggestion during conversion between Hangul and Hanja. Read/write **Boolean**.

expression.**HangulHanjaFastConversion**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example asks the user whether to set Microsoft Word to use fast conversion during conversion between Hangul and Hanja.

```
x = MsgBox("Use fast conversion?", vbYesNo)
If x = vbYes Then
    Options.HangulHanjaFastConversion = True
Else
    Options.HangulHanjaFastConversion = False
End If
```



HasChildShapeRange Property

True if the selection contains child shapes. Read-only **Boolean**.

expression.**HasChildShapeRange**

expression Required. An expression that returns a [Selection](#) object.

Example

This example creates a new document with a drawing canvas, populates the drawing canvas with shapes, and then, after checking that the shapes are child shapes, fills the child shapes with a pattern.

```
Sub ChildShapes()  
    Dim docNew As Document  
    Dim shpCanvas As Shape  
  
    'Create a new document with a drawing canvas and shapes  
    Set docNew = Documents.Add  
    Set shpCanvas = docNew.Shapes.AddCanvas( _  
        Left:=100, Top:=100, Width:=200, Height:=200)  
    shpCanvas.CanvasItems.AddShape msoShapeRectangle, _  
        Left:=0, Top:=0, Width:=100, Height:=100  
    shpCanvas.CanvasItems.AddShape msoShapeOval, _  
        Left:=0, Top:=50, Width:=100, Height:=100  
    shpCanvas.CanvasItems.AddShape msoShapeDiamond, _  
        Left:=0, Top:=100, Width:=100, Height:=100  
  
    'Select all shapes in the canvas  
    shpCanvas.CanvasItems.SelectAll  
  
    'Fill canvas child shapes with a pattern  
    If Selection.HasChildShapeRange = True Then  
        Selection.ChildShapeRange.Fill.Patterned msoPatternDivot  
    Else  
        MsgBox "This is not a range of child shapes."  
    End If  
  
End Sub
```



↳ [Show All](#)

HasDiagram Property

MsoTrue if a shape is a diagram. Read-only [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse Returned if a shape is not a diagram.

msoTriStateMixed Not used for this property.

msoTriStateToggle Not used for this property.

msoTrue Returned if a shape is a diagram.

expression.**HasDiagram**

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

Example

This example searches the current document for diagrams with nodes and if it finds both, creates a black balloon with bold white text.

```
Sub HasDiagramProperties()  
    Dim shpDiagram As Shape  
    Dim shpNode As DiagramNode  
    Dim shpBalloon As Shape  
    Dim docThis As Document  
  
    Set docThis = ThisDocument  
  
    'Look through the current document and if a diagram with one  
    'or more diagram nodes exists, create a balloon with text  
    For Each shpDiagram In docThis.Shapes  
        If shpDiagram.HasDiagram = msoTrue And _  
            shpDiagram.HasDiagramNode = msoTrue Then  
            Set shpBalloon = docThis.Shapes.AddShape _  
                (Type:=msoShapeBalloon, Left:=350, _  
                Top:=75, Width:=150, Height:=150)  
            With shpBalloon  
                With .TextFrame.TextRange  
                    .Text = "This is a diagram with nodes."  
                    .Font.Color = wdColorWhite  
                    .Font.Bold = True  
                    .Font.Name = "Tahoma"  
                    .Font.Size = 15  
                End With  
                .Line.BackColor.RGB = RGB _  
                    (Red:=0, Green:=25, Blue:=25)  
                .Fill.ForeColor.RGB = RGB _  
                    (Red:=0, Green:=25, Blue:=25)  
            End With  
        End If  
    Next shpDiagram  
End Sub
```



↳ [Show All](#)

HasDiagramNode Property

MsoTrue if a shape is a diagram node. Read-only [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse Returned if a shape is not a diagram node.

msoTriStateMixed Not used for this property.

msoTriStateToggle Not used for this property.

msoTrue Returned if a shape is a diagram node.

expression.**HasDiagramNode**

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

Example

This example searches the current document for diagrams with nodes and, if it finds both, creates a black balloon with bold white text.

```
Sub HasDiagramProperties()  
    Dim shpDiagram As Shape  
    Dim shpNode As DiagramNode  
    Dim shpBalloon As Shape  
    Dim docThis As Document  
  
    Set docThis = ThisDocument  
  
    'Looks through the current document and when it finds a diagram  
    ' with one or more diagram nodes, creates a balloon with text  
    For Each shpDiagram In docThis.Shapes  
        If shpDiagram.HasDiagram = msoTrue _  
            And shpDiagram.HasDiagramNode = msoTrue Then  
            Set shpBalloon = docThis.Shapes.AddShape( _  
                Type:=msoShapeBalloon, Left:=350, _  
                Top:=75, Width:=150, Height:=150)  
            With shpBalloon  
                With .TextFrame.TextRange  
                    .Text = "This is a diagram with nodes."  
                    .Font.Color = wdColorWhite  
                    .Font.Bold = True  
                    .Font.Name = "Tahoma"  
                    .Font.Size = 15  
                End With  
                .Line.BackColor.RGB = RGB( _  
                    Red:=0, Green:=25, Blue:=25)  
                .Fill.ForeColor.RGB = RGB( _  
                    Red:=0, Green:=25, Blue:=25)  
            End With  
        End If  
    Next shpDiagram  
End Sub
```



HasFile Property

-

True if the specified subdocument has been saved to a file. Read-only **Boolean**.

Example

This example displays the file name of each subdocument in the active document. The example also displays a message for each subdocument that hasn't been saved.

```
Dim subLoop As Subdocument

For Each subLoop In ActiveDocument.Subdocuments
    subLoop.Range.Select
    If subLoop.HasFile = True Then
        MsgBox subLoop.Path & Application.PathSeparator _
            & subLoop.Name
    Else
        MsgBox "This subdocument has not been saved."
    End If
Next subLoop
```



HasHorizontal Property

-

True if a horizontal border can be applied to the object. Read-only **Boolean**.

Remarks

Horizontal borders can be applied to ranges that contain cells in two or more rows of a table or ranges that contain two or more paragraphs.

Example

This example applies single-line horizontal borders, if the selection supports horizontal borders.

```
If Selection.Borders.HasHorizontal = True Then
    Selection.Borders(wdBorderHorizontal).LineStyle = _
        wdLineStyleSingle
End If
```



HasPassword Property

-

True if a password is required to open the specified document. Read-only **Boolean**.

Example

This example sets the password "kittycat" for the active document and then displays a confirmation message.

```
ActiveDocument.Password = "kittycat"  
If ActiveDocument.HasPassword = True Then _  
    MsgBox "The password is set."
```



HasRoutingSlip Property

-

True if the specified document has a routing slip attached to it. Setting this property to **True** creates a routing slip; setting it to **False** deletes the routing slip. Read/write **Boolean**.

Example

This example removes the routing slip from Sales 1995.doc.

```
Documents("Sales 1995.doc").HasRoutingSlip = False
```

If the active document has a routing slip attached to it, this example routes the document.

```
If ActiveDocument.HasRoutingSlip = True Then  
    ActiveDocument.Route  
End If
```



HasText Property

-

True if the specified shape has text associated with it. Read-only **Boolean**.

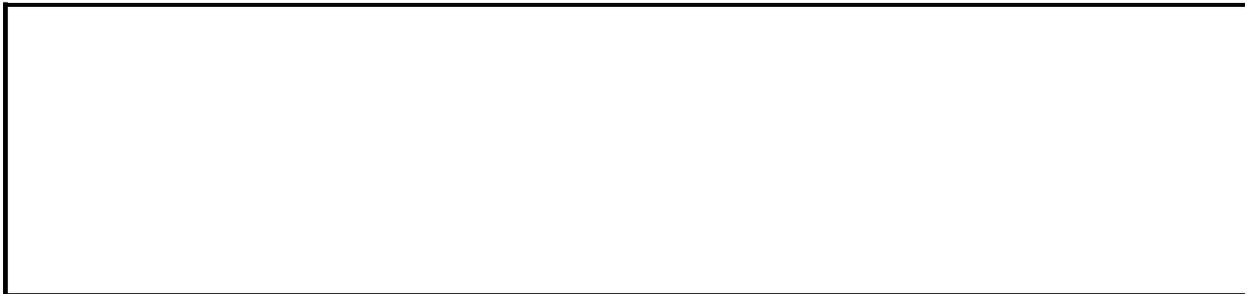
Example

If the second shape on the active document contains text, this example displays a message if the text overflows its frame.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes(2).TextFrame
    If .HasText = True Then
        If .Overflowing = True Then
            MsgBox "Text overflows the frame."
        End If
    End If
End With
```



HasVertical Property

-
True if a vertical border can be applied to the specified object. Read-only **Boolean**.

Remarks

Vertical borders can be applied to ranges that contain cells in two or more columns of a table.

Example

If the selection supports vertical borders, this example applies a single vertical border.

```
If Selection.Borders.HasVertical = True Then  
    Selection.Borders(wdBorderVertical).LineStyle = _  
        wdLineStyleSingle  
End If
```



HeaderDistance Property

-
Returns or sets the distance (in points) between the header and the top of the page. Read/write **Single**.

Example

This example displays the distance between the header and the top of the page. The **PointsToInches** method is used to convert points to inches.

```
Dim sngDistance As Single
```

```
sngDistance = ActiveDocument.PageSetup.HeaderDistance  
Msgbox PointsToInches(sngDistance) & " inches"
```



HeaderFooter Property

Returns a [HeaderFooter](#) object for the specified selection or range. Read-only.

Note An error occurs if the selection isn't located within a header or footer.

Example

This example adds a centered page number to the current page footer.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .SeekView = wdSeekCurrentPageFooter
End With
Selection.HeaderFooter.PageNumbers.Add _
    PageNumberAlignment:=wdAlignPageNumberCenter
```



Headers Property

Returns a [HeadersFooters](#) collection that represents the headers for the specified section. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Remarks

To return a **HeadersFooters** collection that represents the footers for the specified section, use the [Footers](#) property.

Example

This example adds centered page numbers to every page in the active document except the first. (A separate header is created for the first page.)

```
With ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary)
    .PageNumbers.Add _
        PageNumberAlignment:=wdAlignPageNumberCenter, _
        FirstPage:=False
End With
```

This example adds text to the first-page header in the active document.

```
ActiveDocument.PageSetup.DifferentFirstPageHeaderFooter = True
With ActiveDocument.Sections(1).Headers(wdHeaderFooterFirstPage)
    .Range.InsertAfter("First Page Text")
    .Range.Paragraphs.Alignment = wdAlignParagraphRight
End With
```



HeaderSourceName Property

-

Returns the path and file name of the header source attached to the specified mail merge main document. Read-only **String**.

Example

If a header source is attached to the active document, this example displays the file name.

```
Dim strName As String
```

```
strName = ActiveDocument.MailMerge.DataSource.HeaderSourceName  
If strName <> "" Then MsgBox strName
```

This example opens the header source attached to the active document if the source is a Word document.

```
Dim mmdsTemp As MailMergeDataSource
```

```
Set mmdsTemp = ActiveDocument.MailMerge.DataSource
```

```
If mmdsTemp.HeaderSourceType = wdMergeInfoFromWord Then  
    Documents.Open FileName:=mmdsTemp.HeaderSourceName  
End If
```



HeaderSourceType Property

Returns a value that indicates the way the header source is being supplied for the mail merge operation. Read-only [WdMailMergeDataSource](#).

WdMailMergeDataSource can be one of these WdMailMergeDataSource constants.

wdMergeInfoFromAccessDDE

wdMergeInfoFromMSQueryDDE

wdMergeInfoFromODSO

wdNoMergeInfo

wdMergeInfoFromExcelDDE

wdMergeInfoFromODBC

wdMergeInfoFromWord

expression.**HeaderSourceType**

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

Example

This example opens the header source attached to the active document if the source is a Word document.

```
Dim mmdsTemp As MailMergeDataSource  
  
Set mmdsTemp = ActiveDocument.MailMerge.DataSource  
  
If mmdsTemp.HeaderSourceType = wdMergeInfoFromWord Then  
    Documents.Open FileName:=mmdsTemp.HeaderSourceName  
End If
```



HeadingFormat Property

-

True if the specified row or rows are formatted as a table heading. Rows formatted as table headings are repeated when a table spans more than one page. Can be **True**, **False** or **wdUndefined**. Read/write **Long**.

Example

This example creates a 5x5 table at the beginning of the active document and then adds the table heading format to the first table row.

```
Dim rngTemp As Range
Dim tableNew As Table

Set rngTemp = ActiveDocument.Range(0, 0)
Set tableNew = ActiveDocument.Tables.Add(rngTemp, 5, 5)

tableNew.Rows(1).HeadingFormat = True
```

This example determines whether the row that contains the insertion point is formatted as a table heading.

```
If Selection.Information(wdWithInTable) = True Then
    If Selection.Rows(1).HeadingFormat = True Then _
        MsgBox "The current row is a table heading"
Else
    MsgBox "The insertion point is not in a table."
End If
```



HeadingLevelForChapter Property

-

Returns or sets the heading level style that's applied to the chapter titles in the document. Can be a number from 0 (zero) through 8, corresponding to heading levels 1 through 9. Read/write **Long**.

Remarks

Before you can create page numbers that include chapter numbers, the document headings must have a numbered outline format applied that uses styles from the **Bullets and Numbering** dialog box. To do this in Visual Basic, use the **ApplyListTemplate** method.

Example

The first part of this example creates a new document, adds chapter titles and page breaks, and then formats the document by using the last numbered outline format listed in the **Bullets and Numbering** dialog box. The second part of the example adds centered page numbers - including the chapter number - to the header; an en dash separates the chapter number and the page number. The first heading level is used for the chapter number, and lowercase roman numerals are used for the page number.

```
Dim intLoop As Integer
Dim hdrTemp As HeaderFooter

Documents.Add
For intLoop = 1 To 5
    With Selection
        .TypeParagraph
        .InsertBreak
    End With
Next intLoop
ActiveDocument.Content.Style = wdStyleHeading1
ActiveDocument.Content.ListFormat.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(7)

Set hdrTemp = ActiveDocument.Sections(1) _
    .Headers(wdHeaderFooterPrimary)

With hdrTemp.PageNumbers
    .Add PageNumberAlignment:=wdAlignPageNumberCenter
    .NumberStyle = wdPageNumberStyleArabic
    .IncludeChapterNumber = True
    .HeadingLevelForChapter = 0
    .ChapterPageSeparator = wdSeparatorEnDash
End With
```



HeadingSeparator Property

Returns or sets the text between alphabetic groups (entries that start with the same letter) in the index. Corresponds to the \h switch for an INDEX field. Read/write [WdHeadingSeparator](#).

WdHeadingSeparator can be one of these WdHeadingSeparator constants.

wdHeadingSeparatorBlankLine

wdHeadingSeparatorLetterFull

wdHeadingSeparatorNone

wdHeadingSeparatorLetter

wdHeadingSeparatorLetterLow

expression.**HeadingSeparator**

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

Example

This example formats the first index for the active document in a single column, with the appropriate letter preceding each alphabetic group.

```
If ActiveDocument.Indexes.Count >= 1 Then
    With ActiveDocument.Indexes(1)
        .HeadingSeparator = wdHeadingSeparatorLetter
        .NumberOfColumns = 1
    End With
End If
```



HeadingStyles Property

-

Returns a [HeadingStyles](#) object that represents additional styles used to compile a table of contents or table of figures (styles other than the Heading 1 – Heading 9 styles). Read-only.

Example

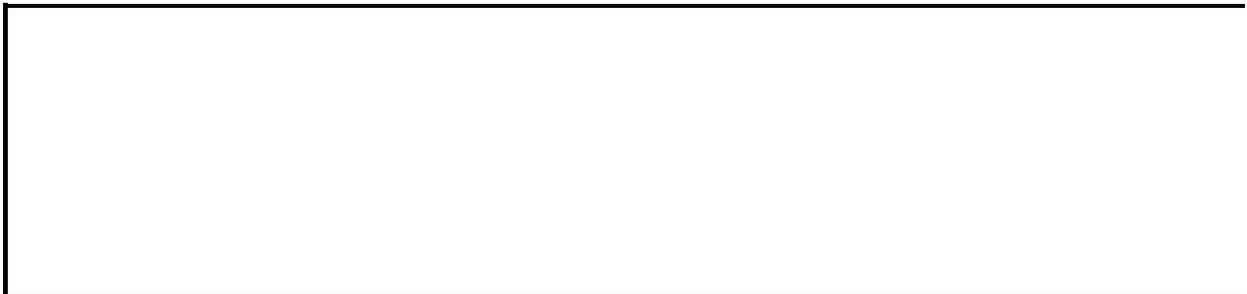
This example adds a style to the **HeadingStyles** collection and then displays the names of all the style in the collection.

```
Dim hsLoop As HeadingStyle

If ActiveDocument.TablesOfContents.Count >=1 Then
    ActiveDocument.TablesOfContents(1).HeadingStyles.Add _
        Style:="Title", Level:=2
    For Each hsLoop In _
        ActiveDocument.TablesOfContents(1).HeadingStyles
        MsgBox hsLoop.Style
    Next hsLoop
End If
```

This example adds a style named "Blue" to the **HeadingStyles** collection in a table of contents for Sales.doc.

```
With Documents("Sales.doc")
    .Styles.Add Name:="Blue"
    .TablesOfContents(1).UseHeadingStyles = True
    .TablesOfContents(1).HeadingStyles.Add _
        Style:="Blue", Level:=4
End With
```



↳ [Show All](#)

HebrewMode Property

-

Returns or sets the mode for the Hebrew spelling checker. Read/write [WdHebSpellStart](#).

WdHebSpellStart can be one of these WdHebSpellStart constants.

wdFullScript The spelling checker follows rules for the conventional script required by the Hebrew Language Academy for writing text without diacritics.

wdMixedAuthorizedScript The spelling checker follows rules for full and partial script, but highlights as potential mistakes any spelling variations not permitted within either system and any completely unrecognized words.

wdMixedScript The spelling checker follows rules for full and partial script and allows non-conventional spelling variations. Only completely unrecognized words are highlighted as potential mistakes.

wdPartialScript The spelling checker follows rules for the traditional script used only for text with diacritics.

expression.**HebrewMode**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the spelling checker to check spelling based on the conventional script required by the Hebrew Language Academy for writing text with diacritics.

```
Options.HebrewMode = wdFullScript
```



↳ [Show All](#)

Height Property

Returns or sets the height of the specified object (in points unless otherwise noted), as shown in the following table.

Object	Height
Application	Returns or sets the height of the active document window. Read/write Long .
Cell, Cells	Returns or sets the height of the specified cell or cells in a table. If the HeightRule property of the specified row is wdRowHeightAuto , Height returns wdUndefined ; setting the Height property sets HeightRule to wdRowHeightAtLeast . Read/write Single .
CustomLabel	Returns or sets the height of the specified custom mailing label. Read/write Single .
Frame	Returns or sets the height of the specified frame. Read/write Single .
Frameset	Returns or sets the height of the specified Frameset object. Read/write Float . The HeightType property determines the type of unit in which this value is expressed.
InlineShape	Returns or sets the height of the specified inline shape. Read/write Single .
Row, Rows	Returns or sets the height of the specified row or rows in a table. If the HeightRule property of the specified row is wdRowHeightAuto , Height returns wdUndefined ; setting the Height property sets HeightRule to wdRowHeightAtLeast . Read/write Single .
Shape, ShapeRange	Returns or sets the height of the specified shape. Read/write Single .
Task	Returns or sets the height of the specified task window. Read/write Long .
	Returns or sets the height of the window. You cannot set this property if the window is maximized or minimized.

Window

Use the [UsableHeight](#) property of the **Application** object to determine the maximum size for the window. Use the [WindowState](#) property to determine the window state. Read/write **Long**.

Example

▶ [As it applies to the **Rows** object.](#)

This example sets the height of the rows in the first table in the active document to at least 20 points.

```
ActiveDocument.Tables(1).Rows.Height = 20
```

▶ [As it applies to the **Row** object.](#)

This example displays the height (in points) of the table row that contains the insertion point.

```
If Selection.Information(wdWithInTable) = True Then
    MsgBox Selection.Rows(1).Height
End If
```

▶ [As it applies to the **Window** object.](#)

This example changes the height of the active window to fill the application window area.

```
With ActiveDocument.ActiveWindow
    .WindowState = wdWindowStateNormal
    .Height = Application.UsableHeight
End With
```

▶ [As it applies to the **ShapeRange** object.](#)

This example inserts a picture as an inline shape and changes the height and width of the image.

```
Set aInLine = _
    ActiveDocument.InlineShapes.AddPicture( _
        FileName:="C:\Windows\Bubbles.bmp", _
        Range:=Selection.Range)
With aInLine
    .Height = 100
    .Width = 200
End With
```

► [As it applies to the **Frameset** object.](#)

This example sets the height of the specified **Frameset** object to 25% of the window height.

```
With ActiveWindow.ActivePane.Frameset
    .HeightType = wdFramesetSizeTypePercent
    .Height = 25
End With
```



↳ [Show All](#)

HeightRule Property

▶ [HeightRule property as it applies to the **Frame** object.](#)

Returns or sets the rule for determining the height of the specified frame.

Read/write [WdFrameSizeRule](#).

WdFrameSizeRule can be one of these WdFrameSizeRule constants.

wdFrameAtLeast

wdFrameExact

wdFrameAuto

expression.**HeightRule**

expression Required. An expression that returns a [Frame](#) object.

▶ [HeightRule property as it applies to the **Cell**, **Cells**, **Row**, and **Rows** objects.](#)

Returns or sets the rule for determining the height of the specified cells or rows.

Read/write [WdRowHeightRule](#).

WdRowHeightRule can be one of these WdRowHeightRule constants.

wdRowHeightAtLeast

wdRowHeightExactly

wdRowHeightAuto

expression.**HeightRule**

expression Required. An expression that returns one of the above objects.

Remarks

Setting the **HeightRule** property of a **Cell** or **Cells** object automatically sets the property for the entire row.

Example

▶ [As it applies to the **Frame** object.](#)

This example sets both the height and width of the first frame in the active document to exactly 1 inch.

```
If ActiveDocument.Frames.Count >= 1 Then
    With ActiveDocument.Frames(1)
        .HeightRule = wdFrameExact
        .Height = InchesToPoints(1)
        .WidthRule = wdFrameExact
        .Width = InchesToPoints(1)
    End With
End If
```

▶ [As it applies to the **Row** object.](#)

This example creates a 3x3 table in a new document and then sets a minimum row height of 24 points for the second row.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=3)
With myTable.Rows(2)
    .Height = 24
    .HeightRule = wdRowHeightAtLeast
End With
```

▶ [As it applies to the **Rows** object.](#)

This example sets the height rule for the selected rows to automatically adjust to the tallest cell in the row.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Rows.HeightRule = wdRowHeightAuto
Else
    MsgBox "The insertion point is not in a table."
End If
```



HeightType Property

Returns or sets the width type for the specified frame on a frames page. Read/write [WdFramesetSizeType](#).

WdFramesetSizeType can be one of these WdFramesetSizeType constants.

wdFramesetSizeTypePercent Microsoft Word interprets the height of the specified frame as a percentage of the screen width.

wdFramesetSizeTypeFixed Word interprets the height of the specified frame as a fixed value (in points).

wdFramesetSizeTypeRelative Word interprets the height of the specified frame relative to the width of other frames on the same frames page.

expression.**HeightType**

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

Example

This example sets the height of the first **Frameset** object in the specified frames page to 25 percent of the window height.

```
With ActiveDocument.ActiveWindow.Panes(1).Frameset
    .HeightType = wdFramesetSizeTypePercent
    .Height = 25
End With
```



HelpText Property

Returns or sets the text that's displayed in a message box when the form field has the focus and the user presses F1. If the **OwnHelp** property is set to **True**, **HelpText** specifies the text string value. If **OwnHelp** is set to **False**, **HelpText** specifies the name of an AutoText entry that contains help text for the form field. Read/write **String**.

Example

This example sets the help text for the form field named "Name."

```
With ActiveDocument.FormFields("Name")  
    .OwnHelp = True  
    .HelpText = "Type your full legal name."  
End With
```

A large empty rectangular box with a black border, representing a form field. It is positioned below the code block and occupies a significant portion of the lower half of the page.

↳ [Show All](#)

Hidden Property

▶ [Hidden property as it applies to the **Style** object.](#)

True if the font is formatted as hidden text. Read/write **Boolean**.

expression.**Hidden**

expression Required. An expression that returns one of the above objects.

▶ [Hidden property as it applies to the **Font** object.](#)

True if the font is formatted as hidden text. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**Hidden**

expression Required. An expression that returns one of the above objects.

Remarks

To control the display of hidden text, use the [ShowHiddenText](#) property of the **View** object.

To control whether properties and methods that return **Range** objects include or exclude hidden text when hidden text isn't displayed, use the [IncludeHiddenText](#) property of the **TextRetrievalMode** object.

Example

▶ [As it applies to the **Font** object.](#)

This example inserts text and formats the password number as hidden text.

```
Selection.Collapse Direction:=wdCollapseEnd
With Selection.Range
    .InsertAfter "Smith account password: 8116"
    .Words(5).Font.Hidden = True
End With
```

This example checks the selection for hidden text.

```
If Selection.Type = wdSelectionNormal Then
    If Selection.Font.Hidden = wdUndefined or _
        Selection.Font.Hidden = True Then
        MsgBox "There's hidden text in the selection."
    Else
        MsgBox "No hidden text in the selection."
    End If
Else
    MsgBox "You need to select some text."
End If
```

This example makes all hidden text in the active window visible and then formats the selection as hidden text.

```
ActiveDocument.ActiveWindow.View.ShowHiddenText = True
If Selection.Type = wdSelectionNormal Then _
    Selection.Font.Hidden = True
```



HidePageNumbersInWeb Property

-

Returns or sets whether page numbers in a table of contents or a table of figures should be hidden when publishing to the Web. Read/write **Boolean**.

Example

This example hides page numbers in the first table of contents if the document is to be published to the Web.

```
ActiveDocument.TableOfContents(1).HidePageNumbersInWeb = True
```



Highlight Property

-
Find object: **True** if highlight formatting is included in the find criteria. Can return or be set to **True**, **False**, or **wdUndefined**. Read/write **Long**.

Note The **wdUndefined** value can be used with the **Find** object to ignore the state of highlight formatting in the selection or range that is searched.

Replacement object: **True** if highlight formatting is applied to the replacement text. Can return or be set to **True**, **False**, or **wdUndefined**. Read/write **Long**.

Example

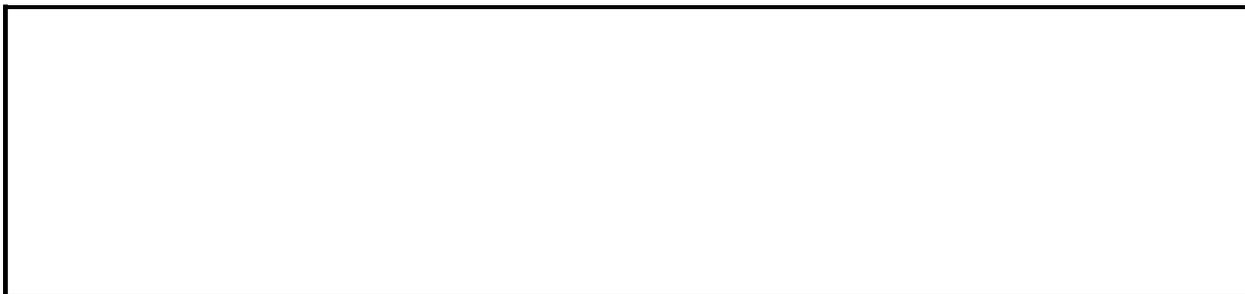
This example finds all instances of highlighted text in the active document and removes the highlight formatting by setting the **Highlight** property of the **Replacement** object to **False**.

```
Dim rngTemp As Range

Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
With rngTemp.Find
    .ClearFormatting
    .Highlight = True
    With .Replacement
        .ClearFormatting
        .Highlight = False
    End With
    .Execute Replace:=wdReplaceAll, Forward:=True, FindText:="", _
        ReplaceWith:="", Format:=True
End With
```

This example applies highlight formatting to the next instance of bold text in the active document.

```
With Selection.Find
    .ClearFormatting
    .Font.Bold = True
    With .Replacement
        .ClearFormatting
        .Highlight = True
    End With
    .Execute Forward:=True, FindText:="", ReplaceWith:="", _
        Format:=True
End With
```



HighlightColorIndex Property

Returns or sets the highlight color for the specified range. Read/write [WdColorIndex](#).

Applies to one of the following **WdColorIndex** constants.

wdByAuthor

wdAuto

wdNoHighlight

wdBlack

wdBlue

wdBrightGreen

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**HighlightColorIndex**

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

Remark

Setting this property to **wdNoHighlight** removes the highlight color (if any) from the specified range.

Example

This example removes highlight formatting from the selection.

```
Selection.Range.HighlightColorIndex = wdNoHighlight
```

This example applies yellow highlighting to each bookmark in the active document.

```
For Each abookmark In ActiveDocument.Bookmarks  
    abookmark.Range.HighlightColorIndex = wdYellow  
Next abookmark
```



HighlightMergeFields Property

True to highlight the merge fields in a document. Read/write **Boolean**.

expression.**HighlightMergeFields**

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

Example

This example turns off highlighting merge fields in the active document.

```
Sub HighlightFields()  
    ActiveDocument.MailMerge.HighlightMergeFields = False  
End Sub
```



HorizontalDistanceFromText Property

Returns or sets the horizontal distance between a frame and the surrounding text, in points. Read/write **Single**.

Example

This example adds a frame around the selection and sets the horizontal distance between the frame and the text to 12 points.

```
Dim frmNew As Frame
```

```
Set frmNew = ActiveDocument.Frames.Add(Range:=Selection.Range)  
frmNew.HorizontalDistanceFromText = 12
```

This example adds a frame around the first paragraph and sets several properties of the frame.

```
Dim frmNew As Frame
```

```
Set frmNew = ActiveDocument.Frames.Add _  
    (Range:=ActiveDocument.Paragraphs(1).Range)
```

```
With frmNew  
    .HorizontalDistanceFromText = InchesToPoints(0.25)  
    .VerticalDistanceFromText = InchesToPoints(0.25)  
    .HeightRule = wdFrameAuto  
    .WidthRule = wdFrameAuto  
    .Borders.Enable = False  
End With
```



↳ [Show All](#)

HorizontalFlip Property

-
Indicates that a shape has been flipped horizontally. Read-only [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**HorizontalFlip**

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

Example

This example restores each shape in the active document to its original state if it's been flipped horizontally or vertically.

```
Sub FlipShape()  
    Dim shpFlip As Shape  
    For Each shpFlip In ActiveDocument.Shapes  
        If shpFlip.HorizontalFlip Then shpFlip.Flip msoFlipHorizontal  
        If shpFlip.VerticalFlip Then shpFlip.Flip msoFlipVertical  
    Next  
End Sub
```



↳ [Show All](#)

HorizontalInVertical Property

Returns or sets the formatting for horizontal text set within vertical text.
Read/write [WdHorizontalInVerticalType](#).

WdHorizontalInVerticalType can be one of these WdHorizontalInVerticalType constants.

wdHorizontalInVerticalNone

wdHorizontalInVerticalFitInLine

wdHorizontalInVerticalResizeLine

expression.**HorizontalInVertical**

expression Required. An expression that returns a [Range](#) object.

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example formats the current selection as horizontal text within a run of vertical text, fitting the text to the line width of the vertical text.

```
Selection.Range.HorizontalInVertical = wdHorizontalInVerticalFitInLi
```



HorizontalLineFormat Property

-

Returns a [HorizontalLineFormat](#) object that contains the horizontal line formatting for the specified **InlineShape** object. Read-only.

Example

This example sets the length of the specified horizontal line to 50% of the window width.

```
ActiveDocument.InlineShapes(1).HorizontalLineFormat _  
    .PercentWidth = 50
```



HorizontalPercentScrolled Property

-

Returns or sets the horizontal scroll position as a percentage of the document width. Read/write **Long**.

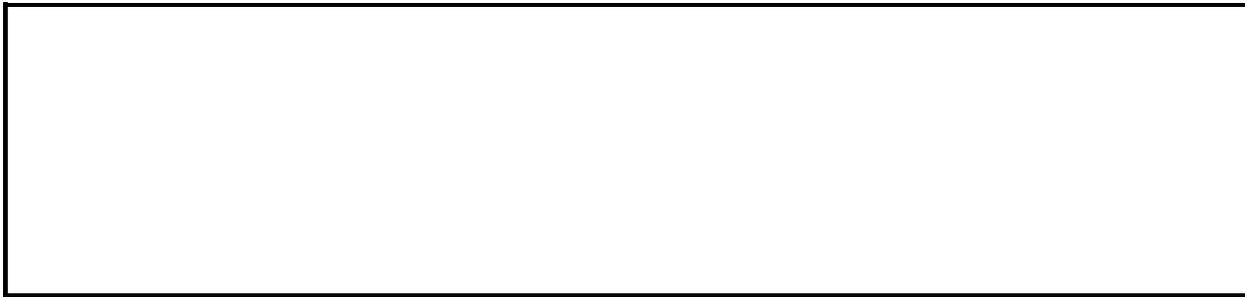
Example

This example displays the percentage that the active window is scrolled horizontally.

```
MsgBox _  
    ActiveDocument.ActiveWindow.HorizontalPercentScrolled & "%"
```

This example vertically scrolls the active pane of the window for Document1 all the way to the left.

```
With Windows("Document1")  
    .Activate  
    .ActivePane.HorizontalPercentScrolled = 0  
End With
```



HorizontalPitch Property

Returns or sets the horizontal distance (in points) between the left edge of one custom mailing label and the left edge of the next mailing label. Read/write **Single**.

Note If this property is changed to a value that isn't valid for the specified mailing label layout, an error occurs.

Example

This example defines the layout of an existing custom label named "Laser labels." The horizontal distance between the left edge of one label and the left edge of the next label is set to 4.19 inches.

```
With Application.MailingLabel.CustomLabels("Laser labels")  
    .Height = InchesToPoints(2)  
    .HorizontalPitch = InchesToPoints(4.19)  
    .NumberAcross = 2  
    .NumberDown = 5  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0.16)  
    .TopMargin = InchesToPoints(0.5)  
    .VerticalPitch = InchesToPoints(2)  
    .Width = InchesToPoints(4)  
End With
```



HorizontalPosition Property

Frame object: Returns or sets the horizontal distance between the edge of the frame and the item specified by the [RelativeHorizontalPosition](#) property. Can be a number that indicates a measurement in points, or can be one of the following **WdFramePosition** constants: **wdFrameLeft**, **wdFrameRight**, **wdFrameCenter**, **wdFrameInside**, or **wdFrameOutside**. Read/write **Single**.

Rows object: Returns or sets the horizontal distance between the edge of the rows and the item specified by the [RelativeHorizontalPosition](#) property. Can be a number that indicates a measurement in points, or can be one of the following **WdTablePosition** constants: **wdTableLeft**, **wdTableRight**, **wdTableCenter**, **wdTableInside**, or **wdTableOutside**. Read/write **Single**. This property doesn't have any effect if [WrapAroundText](#) is **False**.

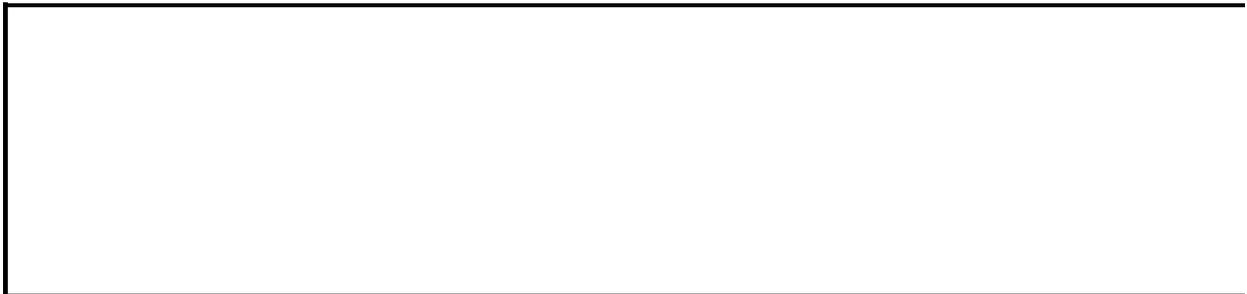
Example

This example aligns the first frame in the active document horizontally with the right margin.

```
If ActiveDocument.Frames.Count >= 1 Then
    With ActiveDocument.Frames(1)
        .RelativeHorizontalPosition = _
            wdRelativeHorizontalPositionMargin
        .HorizontalPosition = wdFrameRight
    End With
End If
```

This example aligns the first table in the active document horizontally with the right margin.

```
If ActiveDocument.Tables.Count >= 1 Then
    With ActiveDocument.Tables(1).Rows
        .RelativeHorizontalPosition = _
            wdRelativeHorizontalPositionMargin
        .HorizontalPosition = wdTableRight
    End With
End If
```



HorizontalResolution Property

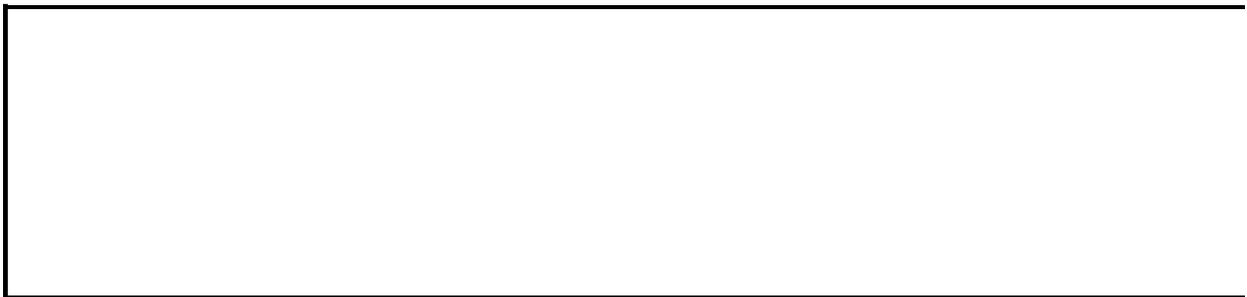
-
Returns the horizontal display resolution, in pixels. Read-only **Long**.

Example

This example displays the current screen resolution (for example, "1024 x 768").

```
Dim lngHorizontal As Long  
Dim lngVertical As Long
```

```
lngHorizontal = System.HorizontalResolution  
lngVertical = System.VerticalResolution  
MsgBox "Resolution = " & lngHorizontal & " x " & lngVertical
```



HTMLDivisions Property

Returns an [HTMLDivisions](#) object that represents an HTML division in a Web document.

expression.**HTMLDivisions**

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

Example

This example formats three nested divisions in the active document. This example assumes that the active document is an HTML document with at least three divisions.

```
Sub FormatHTMLDivisions()  
  With ActiveDocument.HTMLDivisions(1)  
    With .Borders(wdBorderLeft)  
      .Color = wdColorRed  
      .LineStyle = wdLineStyleSingle  
    End With  
    With .Borders(wdBorderRight)  
      .Color = wdColorRed  
      .LineStyle = wdLineStyleSingle  
    End With  
    With .HTMLDivisions(1)  
      .LeftIndent = InchesToPoints(1)  
      .RightIndent = InchesToPoints(1)  
      With .Borders(wdBorderTop)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .Borders(wdBorderBottom)  
        .Color = wdColorBlue  
        .LineStyle = wdLineStyleDouble  
      End With  
      With .HTMLDivisions(1)  
        .LeftIndent = InchesToPoints(1)  
        .RightIndent = InchesToPoints(1)  
        With .Borders(wdBorderLeft)  
          .LineStyle = wdLineStyleDot  
        End With  
        With .Borders(wdBorderRight)  
          .LineStyle = wdLineStyleDot  
        End With  
        With .Borders(wdBorderTop)  
          .LineStyle = wdLineStyleDot  
        End With  
        With .Borders(wdBorderBottom)  
          .LineStyle = wdLineStyleDot  
        End With  
      End With  
    End With  
  End With  
End Sub
```

End With

End Sub



↳ [Show All](#)

HTMLFidelity Property

-
Strips HTML tags used for opening HTML files in Word but not required for display. Read/write [WdEmailHTMLFidelity](#).

WdEmailHTMLFidelity can be one of these WdEmailHTMLFidelity constants.

wdEmailHTMLFidelityHigh Leaves HTML intact.

wdEmailHTMLFidelityLow Removes all HTML tags that do not affect how a message displays.

expression.**HTMLFidelity**

expression Required. An expression that returns an [EmailOptions](#) object.

Example

This example keeps all HTML tags intact when sending e-mail messages.

```
Sub HTMLEmail()  
    Application.EmailOptions _  
        .HTMLFidelity = wdEmailHTMLFidelityHigh  
End Sub
```



HTMLProject Property

Returns an [HTMLProject](#) object in the specified document that represents a top-level project branch, as in the Project Explorer of the Microsoft Script Editor.

expression.**HTMLProject**

expression Required. An expression that returns a [Document](#) object.



Hyperlink Property

Returns a [Hyperlink](#) object that represents the hyperlink associated with the specified **Shape**, **InlineShape**, or **ShapeRange** object. Read-only.

Note If there's no hyperlink associated with the specified shape, an error occurs.

Example

This example displays the address for the hyperlink for the first shape in the active document.

```
MsgBox ActiveDocument.Shapes(1).Hyperlink.Address
```



Hyperlinks Property

-
Returns a [Hyperlinks](#) collection that represents all the hyperlinks in the specified document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the target address of the second hyperlink in Home.doc.

```
If Documents("Home.doc").Hyperlinks.Count >= 2 Then
    MsgBox Documents("Home.doc").Hyperlinks(2).Name
End If
```

This example jumps to the address of the first hyperlink in the selection.

```
If Selection.Hyperlinks.Count >= 1 Then
    Selection.Hyperlinks(1).Follow
End If
```

This example displays the name of every hyperlink in the active document that includes the word "Microsoft" in its address.

```
For Each aHyperlink In ActiveDocument.Hyperlinks
    If InStr(LCase(aHyperlink.Address), "microsoft") <> 0 Then
        MsgBox aHyperlink.Name
    End If
Next aHyperlink
```



HyphenateCaps Property

-

True if words in all capital letters can be hyphenated. Read/write **Boolean**.

Example

This example enables automatic hyphenation for the active document and allows capitalized words to be hyphenated.

```
With ActiveDocument  
    .AutoHyphenation = True  
    .HyphenateCaps = True  
End With
```



Hyphenation Property

-

True if the specified paragraphs are included in automatic hyphenation. **False** if the specified paragraphs are to be excluded from automatic hyphenation. Can be **True**, **False** or **wdUndefined**. Read/write **Long**.

Example

This example turns off automatic hyphenation for all paragraphs in the active document that have the Normal style.

```
ActiveDocument.Styles("Normal").ParagraphFormat.Hyphenation = False
```



HyphenationZone Property

-

Returns or sets the width of the hyphenation zone, in points. The hyphenation zone is the maximum amount of space that Microsoft Word leaves between the end of the last word in a line and the right margin. Read/write **Long**.

Example

This example enables automatic hyphenation for MyReport.doc. The hyphenation zone is set to 36 points (0.5 inch).

```
With Documents("MyReport.doc")  
    .AutoHyphenation = True  
    .HyphenationZone = 36  
End With
```

This example sets the hyphenation zone to 0.25 inch (18 points) and then starts manual hyphenation of the active document.

```
With ActiveDocument  
    .HyphenationZone = InchesToPoints(0.25)  
    .ManualHyphenation  
End With
```



IconIndex Property

Returns or sets the icon that's used when the [DisplayAsIcon](#) property is **True**: 0 (zero) corresponds to the first icon, 1 corresponds to the second icon, and so on. If this argument is omitted, the first (default) icon is used. Read/write **Long**.

expression.**IconIndex**

expression Required. An expression that returns an [OleFormat](#) object.

Example

This example returns the icon index number in a message box for the first selected shape that's displayed as an icon.

```
Dim olefTemp As OLEFormat

If Selection.ShapeRange.Count >= 1 Then
    Set olefTemp = Selection.ShapeRange(1).OLEFormat
    With olefTemp
        If .DisplayAsIcon = True Then MsgBox .IconIndex
    End With
End If
```



IconLabel Property

Returns or sets the text displayed below the icon for an OLE object. Read/write **String**.

expression.**IconLabel**

expression Required. An expression that returns an [OleFormat](#) object.

Example

This example changes the text below the icon for the first shape in the selection.

```
Dim olefTemp As OLEFormat

If Selection.ShapeRange.Count >= 1 Then
    Set olefTemp = Selection.ShapeRange(1).OLEFormat
    With olefTemp
        .DisplayAsIcon = True
        .IconLabel = "My Icon"
    End With
End If
```



IconName Property

Returns or sets the program file in which the icon for an OLE object is stored.
Read/write **String**.

expression.**IconName**

expression Required. An expression that returns an [OleFormat](#) object.

Example

This example changes the first shape in the selection to be displayed as an icon and sets the text below the icon to the icon's file name.

```
Dim olefTemp As OLEFormat

If Selection.ShapeRange.Count >= 1 Then
    Set olefTemp = Selection.ShapeRange(1).OLEFormat
    With olefTemp
        .DisplayAsIcon = True
        .IconLabel = .IconName
    End With
End If
```



IconPath Property

Returns the path of the file in which the icon for an OLE object is stored. Read-only **String**.

expression.**IconPath**

expression Required. An expression that returns an [OleFormat](#) object.

Example

This example displays the path for each embedded OLE object that's displayed as an icon on the active document.

```
Dim shapeLoop As Shape

For Each shapeLoop In ActiveDocument.Shapes
    If shapeLoop.Type = msoEmbeddedOLEObject Then
        If shapeLoop.OLEFormat.DisplayAsIcon = True Then _
            MsgBox shapeLoop.OLEFormat.IconPath
    End If
Next shapeLoop
```



IgnoreInternetAndFileAddresses Property

True if file name extensions, MS-DOS paths, e-mail addresses, server and share names (also known as UNC paths), and Internet addresses (also known as URLs) are ignored while checking spelling. Read/write **Boolean**.

expression.IgnoreInternetAndFileAddresses

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore file names and Internet addresses, and then it checks spelling in the active document.

```
Options.IgnoreInternetAndFileAddresses = True  
ActiveDocument.CheckSpelling
```

This example returns the current status of the **Ignore Internet and file addresses** option on the **Spelling & Grammar** tab in the **Options** dialog box.

```
Dim blnTemp As Boolean
```

```
blnTemp = Options.IgnoreInternetAndFileAddresses
```



IgnoreMixedDigits Property

True if words that contain numbers are ignored while checking spelling.
Read/write **Boolean**.

expression.IgnoreMixedDigits

expression Required. An expression that returns an [Options](#) object.

Example

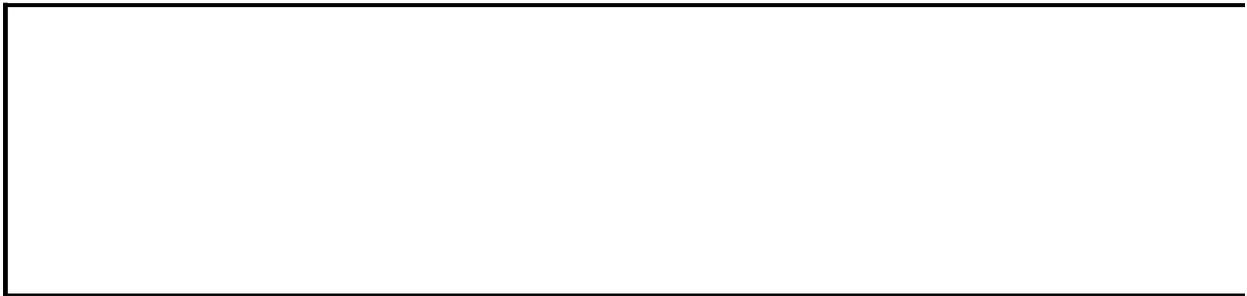
This example sets Microsoft Word to ignore words that contain numbers, and then it checks spelling in the active document.

```
Options.IgnoreMixedDigits = True  
ActiveDocument.CheckSpelling
```

This example returns the current status of the **Ignore words with numbers** option on the **Spelling & Grammar** tab in the **Options** dialog box.

```
Dim blnTemp As Boolean
```

```
blnTemp = Options.IgnoreMixedDigits
```



IgnoreUppercase Property

True if words in all uppercase letters are ignored while checking spelling.
Read/write **Boolean**.

expression.IgnoreUppercase

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to ignore words in all uppercase letters, and then it checks spelling in the active document.

```
Options.IgnoreUppercase = True  
ActiveDocument.CheckSpelling
```

This example returns the current status of the **Ignore words in UPPERCASE** option on the **Spelling & Grammar** tab in the **Options** dialog box.

```
Dim blnTemp As Boolean  
  
blnTemp = Options.IgnoreUppercase
```



IMEAutomaticControl Property

True if Microsoft Word is set to automatically open and close the Japanese Input Method Editor (IME). Read/write **Boolean**.

expression.**IMEAutomaticControl**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to automatically open and close the Japanese Input Method Editor (IME).

```
Options.IMEAutomaticControl = True
```



↳ [Show All](#)

IMEMode Property

Returns or sets the default start-up mode for the Japanese Input Method Editor (IME). Read/write [WdIMEMode](#).

WdIMEMode can be one of these WdIMEMode constants.

wdIMEModeAlpha Activates the IME in half-width Latin mode.

wdIMEModeAlphaFull Activates the IME in full-width Latin mode.

wdIMEModeHangul Activates the IME in half-width Hangul mode.

wdIMEModeHangulFull Activates the IME in full-width Hangul mode.

wdIMEModeHiragana Activates the IME in full-width hiragana mode.

wdIMEModeKatakana Activates the IME in full-width katakana mode.

wdIMEModeKatakanaHalf Activates the IME in half-width katakana mode.

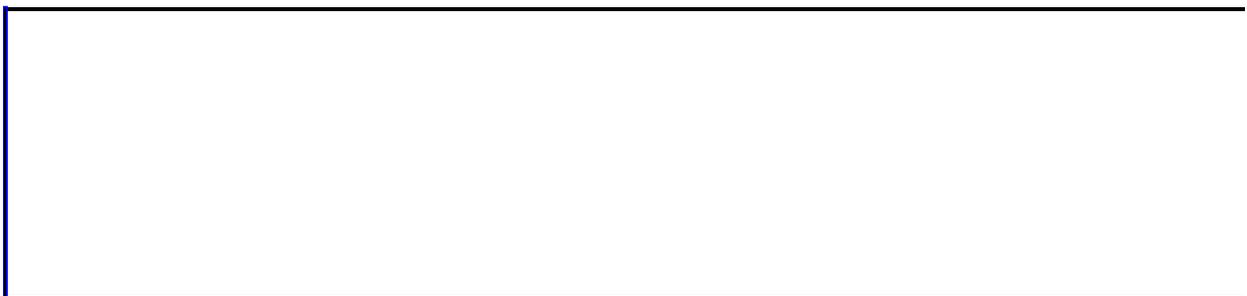
wdIMEModeNoControl Does not change the IME mode.

wdIMEModeOff Disables the IME and activates Latin text entry.

wdIMEModeOn Activates the IME.

expression.**IMEMode**

expression Required. An expression that returns an [Window](#) object.



IncludeCategoryHeader Property

True if the category name for a group of entries appears in the table of authorities. Corresponds to the \h switch for a Table of Authorities (TOA) field. Read/write **Boolean**.

expression.**IncludeCategoryHeader**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example includes the category name for each table of authorities in the active document.

```
Dim toaLoop As TableOfAuthorities  
  
For Each toaLoop In ActiveDocument.TablesOfAuthorities  
    toaLoop.IncludeCategoryHeader = True  
Next toaLoop
```



IncludeChapterNumber Property

True if a chapter number is included with page numbers or a caption label.
Read/write **Boolean**.

expression.**IncludeChapterNumber**

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

Example

This example adds page numbers in the footer for section one in the active document. The page numbers include the chapter number.

```
With ActiveDocument.Sections(1).Footers _  
    (wdHeaderFooterPrimary).PageNumbers  
    .Add  
    .IncludeChapterNumber = True  
    .HeadingLevelForChapter = 1  
End With
```

This example adds the chapter number from the Heading 2 style to figure captions, sets the caption numbering style, and then inserts a new figure caption. The document should already contain a Heading 2 style with numbering.

```
With CaptionLabels(wdCaptionFigure)  
    .IncludeChapterNumber = True  
    .ChapterStyleLevel = 2  
    .NumberStyle = wdCaptionNumberStyleUppercaseLetter  
End With  
Selection.InsertCaption Label:="Figure", Title:=": History"
```



Included Property

-
True if a record is included in a mail merge. Read/write **Boolean**.

expression.**Included**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

Remarks

Use the [SetAllIncludedFlags](#) method to include or exclude all records in a mail merge data source.

Example

This example loops through the records in the mail merge data source and checks if the zip code field (in this case field number six) contains less than five digits. If a record does contain a zip code of less than five digits, the record is excluded from the mail merge and the address is marked as invalid.

```
Sub CheckRecords()  
  
    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 = wdFirstRecord  
    Do  
        intCount = intCount + 1  
  
        'Set the condition that field six must be greater than  
        'or equal to five  
        If Len(.DataFields(6).Value) < 5 Then  
  
            'Exclude the record if field six is less than five  
            .Included = False  
  
            'Mark the record as containing an invalid address fi  
            .InvalidAddress = True  
  
            'Specify the comment attached to the record  
            'explaining 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 = wdNextRecord  
  
    'End the loop when the counter variable equals the
```

```
    'number of records in the data source  
    Loop Until intCount = .RecordCount  
End With
```

```
End Sub
```



IncludeFieldCodes Property

True if the text retrieved from the specified range includes field codes.
Read/write **Boolean**.

Note The default value is the same as the setting of the **Field codes** option on the **View** tab in the **Options** dialog box (**Tools** menu) until this property has been set. Use the [Text](#) property with a [Range](#) object to retrieve text from the specified range.

expression.**IncludeFieldCodes**

expression Required. An expression that returns a [TextRetrievalMode](#) object.

Example

This example displays the text of the first paragraph in the active document in a message box. The example uses the **IncludeFieldCodes** property to exclude field codes.

```
Dim rngTemp As Range

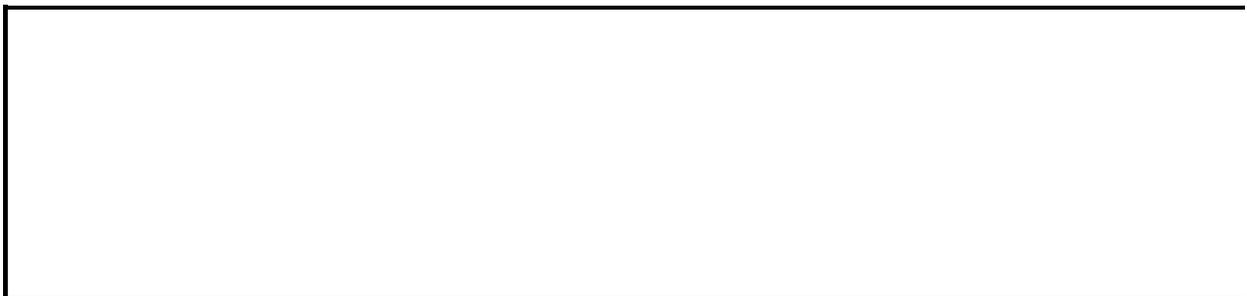
Set rngTemp = ActiveDocument.Paragraphs(1).Range

rngTemp.TextRetrievalMode.IncludeFieldCodes = False
MsgBox rngTemp.Text
```

This example excludes field codes and hidden text from the range that refers to the selected text, and then it displays the text in a message box.

```
Dim rngTemp As Range

If Selection.Type = wdSelectionNormal Then
    Set rngTemp = Selection.Range
    With rngTemp.TextRetrievalMode
        .IncludeHiddenText = False
        .IncludeFieldCodes = False
    End With
    MsgBox rngTemp.Text
End If
```



IncludeHeaderFooter Property

-
True if the header and footer from the page design template are included in a letter created by the Letter Wizard. Read/write **Boolean**.

Note Use the [PageDesign](#) property to set the name of the template attached to a document created by the Letter Wizard.

expression.IncludeHeaderFooter

expression Required. An expression that returns [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, includes the header and footer from the Contemporary Letter template, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim lcNew As LetterContent

Set lcNew = New LetterContent

With lcNew
    .PageDesign = "C:\Program Files\Microsoft Office\" _
        & "Templates\1033\Contemporary Letter.dot"
    .IncludeHeaderFooter = True
End With

Documents.Add.RunLetterWizard LetterContent:=lcNew
```



IncludeHiddenText Property

True if the text retrieved from the specified range includes hidden text.
Read/write **Boolean**.

Note The default value is the same as the current setting of the **Hidden text** option on the **View** tab in the **Options** dialog box (**Tools** menu) until this property has been set. Use the [Text](#) property with a [Range](#) object to retrieve text from the specified range.

expression.**IncludeHiddenText**

expression Required. An expression that returns a [TextRetrievalMode](#) object.

Example

This example displays the text of the first sentence in the active document in a message box. The example uses the **IncludeHiddenText** property to include hidden text.

```
Dim rngTemp As Range  
Set rngTemp = ActiveDocument.Sentences(1)  
rngTemp.TextRetrievalMode.IncludeHiddenText = True  
MsgBox rngTemp.Text
```

This example posts a message if the entire selection is formatted as hidden text.

```
Dim rngTemp As Range  
If Selection.Type = wdSelectionNormal Then  
    Set rngTemp = Selection.Range  
  
    rngTemp.TextRetrievalMode.IncludeHiddenText = False  
    If rngTemp.Text = "" Then MsgBox "Selection is hidden"  
End If
```



IncludeLabel Property

True if the caption label and caption number are included in a table of figures.
Read/write **Boolean**.

expression.**IncludeLabel**

expression Required. An expression that returns a [TableOfFigures](#) object.

Example

This example formats the first table of figures in the active document to exclude caption labels (Figure 1, for example).

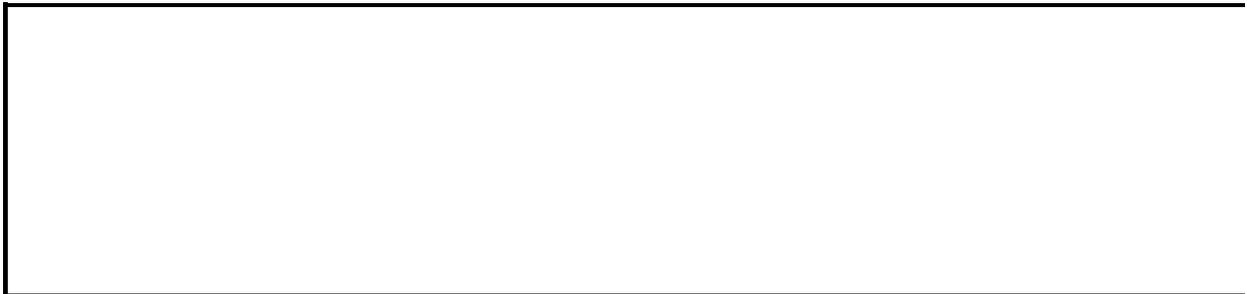
```
If ActiveDocument.TablesOfFigures.Count >= 1 Then
    ActiveDocument.TablesOfFigures(1).IncludeLabel = False
End If
```

This example adds a table of figures in place of the selection and then formats the table to include caption labels.

```
Dim tofTemp As TableOfFigures

Set tofTemp = ActiveDocument.TablesOfFigures _
    .Add(Range:=Selection.Range, _
        Caption:="Figure")

tofTemp.IncludeLabel = True
```



IncludePageNumbers Property

True if page numbers are included in the table of contents or table of figures.
Read/write **Boolean**.

expression.IncludePageNumbers

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

Example

This example formats the first table of contents in the active document to include right-aligned page numbers.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .IncludePageNumbers = True
        .RightAlignPageNumbers = True
    End With
End If
```



IncludeSequenceName Property

Returns or sets the Sequence (SEQ) field identifier for a table of authorities. Corresponds to the \s switch for a Table of Authorities (TOA) field. Read/write **String**.

expression.**IncludeSequenceName**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example inserts a table of authorities at the beginning of the active document and then formats the table to include the Chapter sequence field number before the page number (for example, "Chapter 2-14").

```
Dim rngTemp As Range
Dim toaTemp As TableOfAuthorities

Set rngTemp = ActiveDocument.Range(Start:=0, End:=0)
Set toaTemp = _
    ActiveDocument.TablesOfAuthorities.Add(Range:=rngTemp)

toaTemp.IncludeSequenceName = "Chapter"
```

This example returns the sequence name for the first table of authorities.

```
Dim strSequence As String

strSequence = _
    ActiveDocument.TablesOfAuthorities(1).IncludeSequenceName
```



▾ [Show All](#)

Index Property

▶ [Index property as it applies to the **HeaderFooter** object.](#)

Returns a **WdHeaderFooterIndex** that represents the specified header or footer in a document or section. Read-only.

WdHeaderFooterIndex can be one of these WdHeaderFooterIndex constants.

wdHeaderFooterEvenPages Returns all headers or footers on even-numbered pages.

wdHeaderFooterFirstPage Returns the first header or footer in a document or section.

wdHeaderFooterPrimary Returns the header or footer on all pages other than the first page of a document or section.

expression.**Index**

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

▶ [Index property as it applies to all other objects in the Applies To list.](#)

Returns a **Long** that represents the position of an item in a collection. Read-only.

expression.**Index**

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

Example

▶ [As it applies to the **Field** object.](#)

This example returns the position of the selected field in the **Fields** collection.

```
num = Selection.Fields(1).Index
```

▶ [As it applies to the **HeaderFooter** object.](#)

This example adds a shape to the first page header in the active document if the specified variable references the first page header.

```
Sub ChangeFirstPageFooter()  
    Dim hdrFirstPage As HeaderFooter  
  
    Set hdrFirstPage = ActiveDocument.Sections(1).Headers(wdHeaderFo  
  
    If hdrFirstPage.Index = wdHeaderFooterFirstPage Then  
        With hdrFirstPage.Shapes.AddShape(Type:=msoShapeHeart, _  
            Left:=36, Top:=36, Width:=36, Height:=36)  
            .Fill.ForeColor.RGB = RGB(Red:=255, Green:=0, Blue:=0)  
        End With  
    End If  
  
End Sub
```

▶ [As it applies to the **Variable** object.](#)

This example adds a document variable to the active document and then returns the position of the specified variable in the **Variables** collection.

```
Set myVar = ActiveDocument.Variables.Add(Name:="Name", _  
    Value:="Joe")  
num = myVar.Index
```

▶ [As it applies to the **Window** object.](#)

This example returns the number of the first window in the **Windows** collection. If there are at least two windows in the **Windows** collection, the macro activates the next window, copies the first word, switches back to the original window,

and inserts the Clipboard contents there.

```
Set myWindow = Windows(1)
winNum = myWindow.Index
If Windows.Count >= 2 Then
    myWindow.Next.Activate
    ActiveDocument.Words(1).Copy
    Windows(winNum).Activate
    Selection.Range.Paste
End If
```



Indexes Property

-
Returns an [Indexes](#) collection that represents all the indexes in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

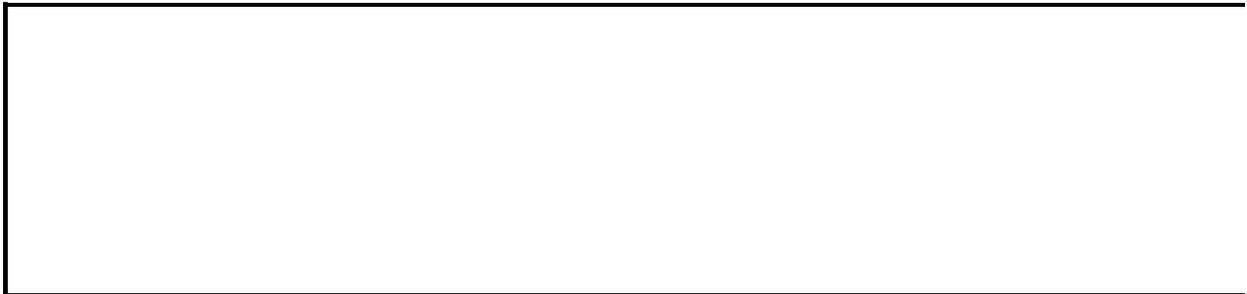
Example

This example adds an index at the end of the active document.

```
Set MyRange = _  
    ActiveDocument.Range(Start:=ActiveDocument.Content.End - 1, _  
        End:=ActiveDocument.Content.End - 1)  
ActiveDocument.Indexes.Add Range:=MyRange, NumberOfColumns:=1, _  
    HeadingSeparator:=False
```

This example inserts an index entry for the selected text.

```
If Selection.Type = wdSelectionNormal Then  
    ActiveDocument.Indexes.MarkEntry Range:=Selection.Range, _  
        Entry:=Selection.Range.Text  
End If
```



↳ [Show All](#)

IndexLanguage Property

Returns or sets the sorting language to use for the specified index. Read/write [WdLanguageID](#).

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdAmharic

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt

wdArabicIraq

wdArabicJordan

wdArabicKuwait

wdArabicLebanon

wdArabicLibya

wdArabicMorocco

wdArabicOman

wdArabicQatar

wdArabicSyria

wdArabicTunisia

wdArabicUAE

wdArabicYemen

wdArmenian

wdAssamese

wdAzeriCyrillic

wdAzeriLatin

wdBasque

wdBelgianDutch
wdBelgianFrench
wdBengali
wdBrazilianPortuguese
wdBulgarian
wdBurmese
wdByelorussian
wdCatalan
wdCherokee
wdChineseHongKong
wdChineseMacao
wdChineseSingapore
wdCroatian
wdCzech
wdDanish
wdDivehi
wdDutch
wdEdo
wdEnglishAUS
wdEnglishBelize
wdEnglishCanadian
wdEnglishCaribbean
wdEnglishIreland
wdEnglishJamaica
wdEnglishNewZealand
wdEnglishPhilippines
wdEnglishSouthAfrica
wdEnglishTrinidad
wdEnglishUK
wdEnglishUS
wdEnglishZimbabwe
wdEstonian
wdFaeroese

wdFarsi
wdFilipino
wdFinnish
wdFrench
wdFrenchCameroon
wdFrenchCanadian
wdFrenchCotedIvoire
wdFrenchLuxembourg
wdFrenchMali
wdFrenchMonaco
wdFrenchReunion
wdFrenchSenegal
wdFrenchWestIndies
wdFrenchZaire
wdFrisianNetherlands
wdFulfulde
wdGaelicIreland
wdGaelicScotland
wdGalician
wdGeorgian
wdGerman
wdGermanAustria
wdGermanLiechtenstein
wdGermanLuxembourg
wdGreek
wdGuarani
wdGujarati
wdHausa
wdHawaiian
wdHebrew
wdHindi
wdHungarian
wdIbibio

wdIcelandic
wdIgbo
wdIndonesian
wdInuktitut
wdItalian
wdJapanese
wdKannada
wdKanuri
wdKashmiri
wdKazakh
wdKhmer
wdKirghiz
wdKonkani
wdKorean
wdKyrgyz
wdLanguageNone
wdLao
wdLatin
wdLatvian
wdLithuanian
wdMacedonian
wdMalayalam
wdMalayBruneiDarussalam
wdMalaysian
wdMaltese
wdManipuri
wdMarathi
wdMexicanSpanish
wdMongolian
wdNepali
wdNoProofing
wdNorwegianBokmol
wdNorwegianNynorsk

wdOriya
wdOromo
wdPashto
wdPolish
wdPortuguese
wdPunjabi
wdRhaetoRomanic
wdRomanian
wdRomanianMoldova
wdRussian
wdRussianMoldova
wdSamiLappish
wdSanskrit
wdSerbianCyrillic
wdSerbianLatin
wdSesotho
wdSimplifiedChinese
wdSindhi
wdSindhiPakistan
wdSinhalese
wdSlovak
wdSlovenian
wdSomali
wdSorbian
wdSpanish
wdSpanishArgentina
wdSpanishBolivia
wdSpanishChile
wdSpanishColombia
wdSpanishCostaRica
wdSpanishDominicanRepublic
wdSpanishEcuador
wdSpanishElSalvador

wdSpanishGuatemala
wdSpanishHonduras
wdSpanishModernSort
wdSpanishNicaragua
wdSpanishPanama
wdSpanishParaguay
wdSpanishPeru
wdSpanishPuertoRico
wdSpanishUruguay
wdSpanishVenezuela
wdSutu
wdSwahili
wdSwedish
wdSwedishFinland
wdSwissFrench
wdSwissGerman
wdSwissItalian
wdSyriac
wdTajik
wdTamazight
wdTamazightLatin
wdTamil
wdTatar
wdTelugu
wdThai
wdTibetan
wdTigrignaEritrea
wdTigrignaEthiopic
wdTraditionalChinese
wdTsonga
wdTswana
wdTurkish
wdTurkmen

wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin
wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdYi
wdYiddish
wdYoruba
wdZulu

expression.**IndexLanguage**

expression Required. An expression that returns an [Index](#) object.

Remarks

Some of these constants 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 sets the sorting language of the first index in the active document to New Zealand English.

```
ActiveDocument.Indexes(1).IndexLanguage = _  
    wdEnglishNewZealand
```



InfoBlock Property

-
Associated with the Letter Wizard in Microsoft Word. Not used in the U.S. English version of Word.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

↳ [Show All](#)

Information Property

Returns information about the specified selection or range. Read-only **Variant**.

expression.**Information**(*Type*)

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

Type Required [WdInformation](#). The information type.

WdInformation can be one of these WdInformation constants.

wdActiveEndAdjustedPageNumber Returns the number of the page that contains the active end of the specified selection or range. If you set a starting page number or make other manual adjustments, returns the adjusted page number (unlike **wdActiveEndPageNumber**).

wdActiveEndPageNumber Returns the number of the page that contains the active end of the specified selection or range, counting from the beginning of the document. Any manual adjustments to page numbering are disregarded (unlike **wdActiveEndAdjustedPageNumber**).

wdActiveEndSectionNumber Returns the number of the section that contains the active end of the specified selection or range.

wdAtEndOfRowMarker Returns **True** if the specified selection or range is at the end-of-row mark in a table.

wdCapsLock Returns **True** if Caps Lock is in effect.

wdEndOfRangeColumnNumber Returns the table column number that contains the end of the specified selection or range.

wdEndOfRangeRowNumber Returns the table row number that contains the end of the specified selection or range.

wdFirstCharacterColumnNumber Returns the character position of the first character in the specified selection or range. If the selection or range is collapsed, the character number immediately to the right of the range or selection is returned (this is the same as the character column number displayed

in the status bar after "Col").

wdFirstCharacterLineNumber Returns the character position of the first character in the specified selection or range. If the selection or range is collapsed, the character number immediately to the right of the range or selection is returned (this is the same as the character column number displayed in the status bar after "Col").

wdFrameIsSelected Returns **True** if the selection or range is an entire frame or text box.

wdHeaderFooterType Returns a value that indicates the type of header or footer that contains the specified selection or range, as shown in the following table.

wdHorizontalPositionRelativeToPage Returns the horizontal position of the specified selection or range; this is the distance from the left edge of the selection or range to the left edge of the page measured in points (1 point = 20 twips, 72 points = 1 inch). If the selection or range isn't within the screen area, returns - 1.

wdHorizontalPositionRelativeToTextBoundary Returns the horizontal position of the specified selection or range relative to the left edge of the nearest text boundary enclosing it, in points (1 point = 20 twips, 72 points = 1 inch). If the selection or range isn't within the screen area, returns - 1.

wdInClipboard For information about this constant, consult the language reference Help included with Microsoft Office Macintosh Edition.

wdInCommentPane Returns **True** if the specified selection or range is in a comment pane.

wdInEndnote Returns **True** if the specified selection or range is in an endnote area in print layout view or in the endnote pane in normal view.

wdInFootnote Returns **True** if the specified selection or range is in a footnote area in print layout view or in the footnote pane in normal view.

wdInFootnoteEndnotePane Returns **True** if the specified selection or range is in the footnote or endnote pane in normal view or in a footnote or endnote area in print layout view. For more information, see the descriptions of **wdInFootnote** and **wdInEndnote** in the preceding paragraphs.

wdInHeaderFooter Returns **True** if the selection or range is in the header or footer pane or in a header or footer in print layout view.

Value	Type of header or footer
- 1	None (the selection or range isn't in a header or footer)

0 (zero)	Even page header
1	Odd page header (or the only header, if there aren't odd and even headers)
2	Even page footer
3	Odd page footer (or the only footer, if there aren't odd and even footers)
4	First page header
5	First page footer

wdInMasterDocument Returns **True** if the selection or range is in a master document (that is, a document that contains at least one subdocument).

wdInWordMail Returns **True** if the selection or range is in the header or footer pane or in a header or footer in print layout view.

Value	Location
0(zero)	The selection or range isn't in an e-mail message.
1	The selection or range is in an e-mail message you are sending.
2	The selection or range is in an e-mail you are reading.

wdMaximumNumberOfColumns Returns the greatest number of table columns within any row in the selection or range.

wdMaximumNumberOfRows Returns the greatest number of table rows within the table in the specified selection or range.

wdNumberOfPagesInDocument Returns the number of pages in the document associated with the selection or range.

wdNumLock Returns **True** if Num Lock is in effect.

wdOverType Returns **True** if Overtyping mode is in effect. The [Overtyping](#) property can be used to change the state of the Overtyping mode.

wdReferenceOfType Returns a value that indicates where the selection is in relation to a footnote, endnote, or comment reference, as shown in the following table.

Value	Description
- 1	The selection or range includes but isn't limited to a footnote, endnote, or comment reference.
0 (zero)	The selection or range isn't before a footnote, endnote, or comment reference.
1	The selection or range is before a footnote reference.

- 2 The selection or range is before an endnote reference.
- 3 The selection or range is before a comment reference.

wdRevisionMarking Returns **True** if change tracking is in effect.

wdSelectionMode Returns a value that indicates the current selection mode, as shown in the following table.

Value	Selection mode
0 (zero)	Normal selection
1	Extended selection ("EXT" appears on the status bar)
2	Column selection. ("COL" appears on the status bar)

wdStartOfRangeColumnNumber Returns the table column number that contains the beginning of the selection or range.

wdStartOfRangeRowNumber Returns the table row number that contains the beginning of the selection or range.

wdVerticalPositionRelativeToPage Returns the vertical position of the selection or range; this is the distance from the top edge of the selection to the top edge of the page measured in points (1 point = 20 twips, 72 points = 1 inch). If the selection isn't visible in the document window, returns - 1.

wdVerticalPositionRelativeToTextBoundary Returns the vertical position of the selection or range relative to the top edge of the nearest text boundary enclosing it, in points (1 point = 20 twips, 72 points = 1 inch). This is useful for determining the position of the insertion point within a frame or table cell. If the selection isn't visible, returns - 1.

wdWithinTable Returns **True** if the selection is in a table.

wdZoomPercentage Returns the current percentage of magnification as set by the [Percentage](#) property.

Example

This example displays the current page number and the total number of pages in the active document.

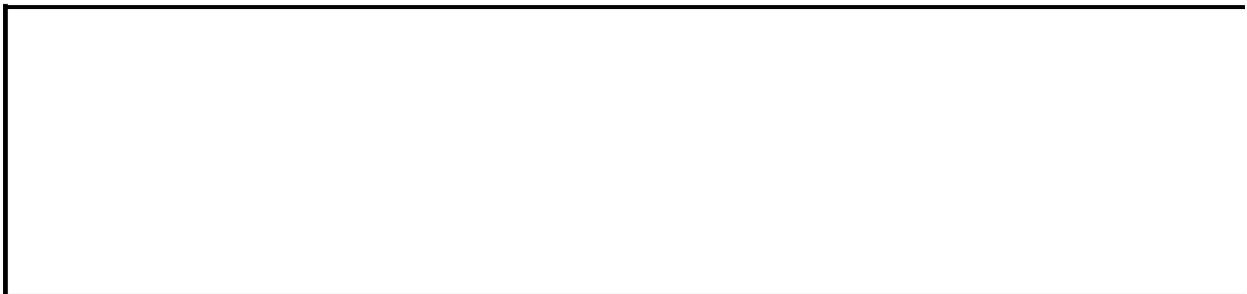
```
MsgBox "The selection is on page " & _  
    Selection.Information(wdActiveEndPageNumber) & " of page " _  
    & Selection.Information(wdNumberOfPagesInDocument)
```

If the selection is in a table, this example selects the table.

```
If Selection.Information(wdWithInTable) Then _  
    Selection.Tables(1).Select
```

This example displays a message that indicates the current section number.

```
Selection.Collapse Direction:=wdCollapseStart  
MsgBox "The insertion point is in section " & _  
    Selection.Information(wdActiveEndSectionNumber)
```



Initial Property

Returns or sets the initials of the user associated with a specific comment.
Read/write **String**.

expression.**Initial**

expression Required. An expression that returns a [Comment](#) object.

Example

This example displays the initials of the user who made the first comment in the selection.

```
If Selection.Comments.Count >= 1 Then
    MsgBox "Comment made by " & Selection.Comments(1).Initial
End If
```

This example checks the author initials associated with each comment in the first document section. If the author initials are "MSOffice," the example changes them to "KAE."

```
Dim rngTemp As Range
Dim comLoop As Comment

Set rngTemp = ActiveDocument.Sections(1).Range
For Each comLoop In rngTemp.Comments
    If comLoop.Initial = "MSOffice" Then comLoop.Initial = "KAE"
Next comLoop
```



Ink Property

Returns or sets a **Single** that represents the degree of saturation for a specified ink. Read/write.

expression.**Ink**(*Index*)

expression Required. An expression that returns a [ColorFormat](#) object.

Index Required **Long**. The number of the ink.

Remarks

The value of the **Ink** property can be any number between 0 and 1. Zero (0) means no ink; one (1) means full saturation. For example, 0.5 would be 50% saturation of the specified ink.

Example

This example creates a new shape in the active document, sets the fill color, and specifies the degree of saturation for two of the four CMYK colors.

```
Sub TintShade()  
  Dim shpHeart As Shape  
  Set shpHeart = ActiveDocument.Shapes.AddShape( _  
    Type:=msoShapeHeart, Left:=150, _  
    Top:=150, Width:=250, Height:=250)  
  With shpHeart.Fill.ForeColor  
    .SetCMYK Cyan:=0, Magenta:=125, Yellow:=12, Black:=25  
    .TintAndShade = 0.3  
    .OverPrint = msoTrue  
    .Ink(Index:=1) = 0.3  
    .Ink(Index:=2) = 0.7  
  End With  
End Sub
```



InlineConversion Property

-
True if Microsoft Word displays an unconfirmed character string in the Japanese Input Method Editor (IME) as an insertion between existing (confirmed) character strings. Read/write **Boolean**.

expression.**InlineConversion**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets Microsoft Word to display an unconfirmed character string in the Japanese Input Method Editor (IME) as an insertion between existing (confirmed) character strings.

```
Options.InlineConversion = True
```



InlineShape Property

Returns an [InlineShape](#) object that represents the picture, OLE object, or ActiveX control that is the result of an INCLUDEPICTURE or EMBED field.

expression.**InlineShape**

expression Required. An expression that returns a [Field](#) object.

Remarks

An **InlineShape** object is treated like a character and is positioned as a character within a line of text.

Example

This example returns the width of the inline shape associated with the first field in the active document. For this example to work, the field must be an INCLUDEPICTURE field.

```
If ActiveDocument.Fields(1).Type = wdFieldIncludePicture Then  
    MsgBox ActiveDocument.Fields(1).InlineShape.Width  
End If
```



InlineShapes Property

Returns an [InlineShapes](#) collection that represents all the **InlineShape** objects in a document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the number of shapes and inline shapes in the active document.

```
Set doc = ActiveDocument
Msgbox "InlineShape = " & doc.InlineShapes.Count & _
      vbCrLf & "Shapes = " & doc.Shapes.Count
```



↳ [Show All](#)

InsertedTextColor Property

Returns or sets the color of text that is inserted while change tracking is enabled.
Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**InsertedTextColor**

expression Required. An expression that returns an [Options](#) object.

Remarks

If the **InsertedTextColor** property is set to **wdByAuthor**, Microsoft Word automatically assigns a unique color to each of the first eight authors who revise a document.

Example

This example sets the color of inserted text to dark red.

```
Options.InsertedTextColor = wdDarkRed
```

This example returns the current status of the **Color** option under **Track Changes options** on the **Track Changes** tab in the **Options** dialog box.

```
Dim lngColor As Long
```

```
lngColor = Options.InsertedTextColor
```



↳ [Show All](#)

InsertedTextMark Property

Returns or sets how Microsoft Word formats inserted text while change tracking is enabled (the [TrackRevisions](#) property is True). If change tracking is not enabled, this property is ignored. Use this property with the [InsertedTextColor](#) property to control the appearance of inserted text in a document. Read/write [WdInsertedTextMark](#).

WdInsertedTextMark can be one of these WdInsertedTextMark constants.

wdInsertedTextMarkBold

wdInsertedTextMarkColorOnly

wdInsertedTextMarkDoubleUnderline

wdInsertedTextMarkItalic

wdInsertedTextMarkNone

wdInsertedTextMarkStrikeThrough

wdInsertedTextMarkUnderline

expression.**InsertedTextMark**

expression Required. An expression that returns an [Options](#) object.

Remarks

The [ShowRevisions](#) property must be **True** in order to see the formatting for inserted text during editing. The [PrintRevisions](#) property must be **True** in order for Word to use the formatting for inserted text when printing a document.

Example

This example sets Word to italicize inserted text.

```
Options.InsertedTextMark = wdInsertedTextMarkItalic
```

This example sets Word to format inserted text as bold if it isn't already.

```
If Options.InsertedTextMark <> wdInsertedTextMarkBold Then  
    Options.InsertedTextMark = wdInsertedTextMarkBold  
Else  
    MsgBox Prompt:="Inserted text is already bold!"  
End If
```



↳ [Show All](#)

InsetPen Property

MsoTrue to draw lines on the inside of a specified shape. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used for this property.

msoFalse Draws lines centered on a shape's border.

msoTriStateMixed Not used for this property.

msoTriStateToggle Not used for this property.

msoTrue Draws lines on the inside of the shapes

expression.**InsetPen**

expression Required. An expression that returns a [LineFormat](#) object.

Remarks

Use the **InsetPen** property to match up the edges of shapes of equal width but whose line widths vary.

Example

This example sets all shapes in the active document to draw lines on the inside of the shapes.

```
Sub InsetLine()  
    Dim shpShape As Shape  
  
    For Each shpShape In ActiveDocument.Shapes  
        shpShape.Line.InsetPen = msoTrue  
    Next shpShape  
End Sub
```



Inside Property

-
True if an inside border can be applied to the specified object. Read-only **Boolean**.

expression.**Inside**

expression Required. An expression that returns a [Border](#) object.

Example

If the current selection supports inside borders (that is, if multiple paragraphs or cells are selected), this example applies a single inside border.

```
Dim borderLoop As Border
```

```
For Each borderLoop In Selection.Borders
```

```
    If borderLoop.Inside = True Then _
```

```
        borderLoop.LineStyle = wdLineStyleSingle
```

```
Next borderLoop
```



↳ [Show All](#)

InsideColor Property

Returns or sets the 24-bit color of the inside borders. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function. Read/write.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic

wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen
wdColorLightTurquoise

wdColorLime
wdColorOrange
wdColorPink

expression.**InsideColor**

expression Required. An expression that returns a [Border](#) object.

Remarks

If the [InsideLineStyle](#) property is set to either `wdLineStyleNone` or `False`, setting this property has no effect.

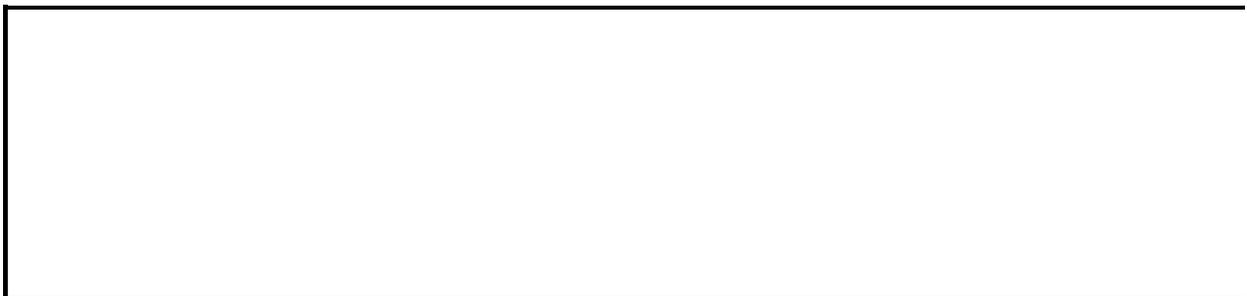
Example

This example adds borders between rows and between columns in the first table of the active document, and then it sets the colors for both the inside and outside borders.

```
If ActiveDocument.Tables.Count >= 1 Then
    Set myTable = ActiveDocument.Tables(1)
    With myTable.Borders
        .InsideLineStyle = True
        .InsideColor = wdColorBlueGray
        .OutsideColor = wdColorPink
    End With
End If
```

This example adds dark red borders between the first four paragraphs in the active document.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range(Start:=myDoc.Paragraphs(1).Range.Start, _
    End:=myDoc.Paragraphs(4).Range.End)
With myRange.Borders
    .InsideLineStyle = wdLineStyleSingle
    .InsideLineWidth = wdLineWidth150pt
    .InsideColor = wdDarkRed
End With
```



↳ [Show All](#)

InsideColorIndex Property

Returns or sets the color of the inside borders. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**InsideColorIndex**

expression Required. An expression that returns a [Border](#) object.

Remarks

If the [InsideLineStyle](#) property is set to either `wdLineStyleNone` or `False`, setting this property has no effect.

Example

This example adds borders between rows and between columns in the first table in the active document, and then it sets the colors for both the inside and outside borders.

```
Dim tableTemp As Table

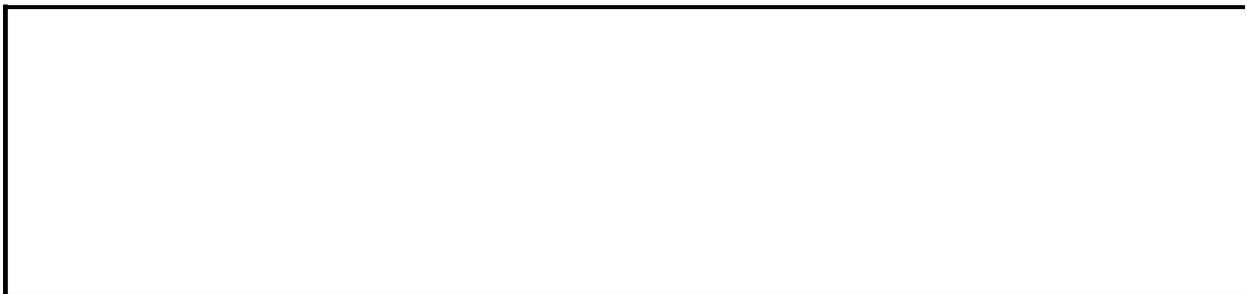
If ActiveDocument.Tables.Count >= 1 Then
    Set tableTemp = ActiveDocument.Tables(1)
    With tableTemp.Borders
        .InsideLineStyle = True
        .InsideColorIndex = wdBrightGreen
        .OutsideColorIndex = wdPink
    End With
End If
```

This example adds red borders between the first four paragraphs in the active document.

```
Dim docActive As Document
Dim rngTemp As Range

Set docActive = ActiveDocument
Set rngTemp = docActive.Range( _
    Start:=docActive.Paragraphs(1).Range.Start, _
    End:=docActive.Paragraphs(4).Range.End)

With rngTemp.Borders
    .InsideLineStyle = wdLineStyleSingle
    .InsideLineWidth = wdLineWidth150pt
    .InsideColorIndex = wdRed
End With
```



↳ [Show All](#)

InsideLineStyle Property

Returns or sets the inside border for the specified object. Returns **wdUndefined** if more than one kind of border is applied to the specified object; otherwise, returns **False** or a **WdLineStyle** constant. Can be set to **True**, **False**, or a [WdLineStyle](#) constant.

WdLineStyle can be one of these WdLineStyle constants.

wdLineStyleDashDot

wdLineStyleDashDotDot

wdLineStyleDashDotStroked

wdLineStyleDashLargeGap

wdLineStyleDashSmallGap

wdLineStyleDot

wdLineStyleDouble

wdLineStyleDoubleWavy

wdLineStyleEmboss3D

wdLineStyleEngrave3D

wdLineStyleInset

wdLineStyleNone

wdLineStyleOutset

wdLineStyleSingle

wdLineStyleSingleWavy

wdLineStyleThickThinLargeGap

wdLineStyleThickThinMedGap

wdLineStyleThickThinSmallGap

wdLineStyleThinThickLargeGap

wdLineStyleThinThickMedGap

wdLineStyleThinThickSmallGap

wdLineStyleThinThickThinLargeGap

wdLineStyleThinThickThinMedGap
wdLineStyleThinThickThinSmallGap
wdLineStyleTriple

expression.**InsideLineStyle**

expression Required. An expression that returns a [Border](#) object.

Remarks

True sets the line style to the default line style and the line width to the default line width. The default line style and line width can be set using the [DefaultBorderLineWidth](#) and [DefaultBorderLineStyle](#) properties.

Use either of the following instructions to remove the inside border from the first table in the active document.

```
ActiveDocument.Tables(1).Borders.InsideLineStyle = wdLineStyleNone  
ActiveDocument.Tables(1).Borders.InsideLineStyle = False
```

Example

This example adds borders between rows and between columns in the first table of the active document.

```
Dim tableTemp As Table

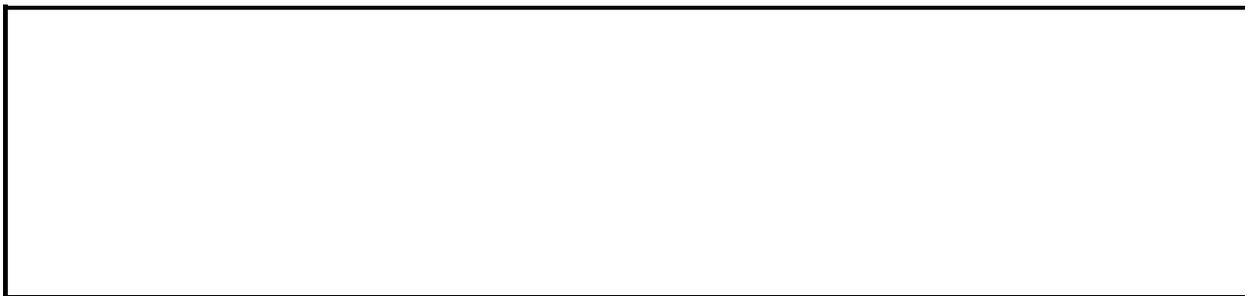
If ActiveDocument.Tables.Count >= 1 Then
    Set tableTemp = ActiveDocument.Tables(1)
    tableTemp.Borders.InsideLineStyle = True
End If
```

This example adds borders between the first four paragraphs in the document.

```
Dim docActive As Document
Dim rngTemp As Range

Set docActive = ActiveDocument
Set rngTemp = docActive.Range( _
    Start:=docActive.Paragraphs(1).Range.Start, _
    End:=docActive.Paragraphs(4).Range.End)

With rngTemp.Borders
    .InsideLineStyle = wdLineStyleSingle
    .InsideLineWidth = wdLineWidth150pt
End With
```



↳ [Show All](#)

InsideLineWidth Property

Returns or sets the line width of the inside border of an object. Returns **wdUndefined** if the object has inside borders with more than one line width; otherwise, returns **False** or a **WdLineWidth** constant. Can be set to **True**, **False**, or one of the following [WdLineWidth](#) constants.

WdLineWidth can be one of these WdLineWidth constants.

wdLineWidth025pt

wdLineWidth050pt

wdLineWidth075pt

wdLineWidth100pt

wdLineWidth150pt

wdLineWidth225pt

wdLineWidth300pt

wdLineWidth450pt

wdLineWidth600pt

expression.**InsideLineWidth**

expression Required. An expression that returns a [Border](#) object.

Example

This example adds borders between rows and between columns in the first table in the active document.

```
Dim tableTemp As Table

If ActiveDocument.Tables.Count >= 1 Then
    Set tableTemp = ActiveDocument.Tables(1)
    tableTemp.Borders.InsideLineStyle = wdLineStyleDot
    tableTemp.Borders.InsideLineWidth = wdLineWidth050pt
End If
```

This example adds dotted borders between the first four paragraphs of the active document.

```
Dim docActive As Document
Dim rngTemp As Range

Set docActive = ActiveDocument
Set rngTemp = docActive.Range( _
    Start:=docActive.Paragraphs(1).Range.Start, _
    End:=docActive.Paragraphs(4).Range.End)

rngTemp.Borders.InsideLineStyle = wdLineStyleDot
rngTemp.Borders.InsideLineWidth = wdLineWidth075pt
```



INSKeyForPaste Property

-
True if the INS key can be used for pasting the Clipboard contents. Read/write **Boolean**.

expression.**INSKeyForPaste**

expression Required. An expression that returns an [Options](#) object.

Example

This example enables the INS key to be used for pasting the contents of the Clipboard.

```
Options.INSKeyForPaste = True
```

This example returns the status of the **Use the INS key for paste** option on the **Edit** tab in the **Options** dialog box.

```
Dim blnTemp As Boolean
```

```
blnTemp = Options.INSKeyForPaste
```



Installed Property

True if the specified add-in is installed (loaded). Add-ins that are loaded are selected in the **Templates and Add-ins** dialog box (**Tools** menu). Read/write **Boolean**.

Note Uninstalled add-ins are included in the **AddIns** collection. To remove a template or WLL from the **AddIns** collection, apply the [Delete](#) method to the **AddIn** object (the add-in name is removed from the **Templates and Add-ins** dialog box). To unload all templates and WLLs, apply the [Unload](#) method to the **AddIns** collection.

expression.**Installed**

expression Required. An expression that returns an [AddIn](#) object.

Example

This example unloads the global template named "Gallery.dot."

```
Addins("Gallery.dot").Installed = False
```

This example loads FindAll.wll.

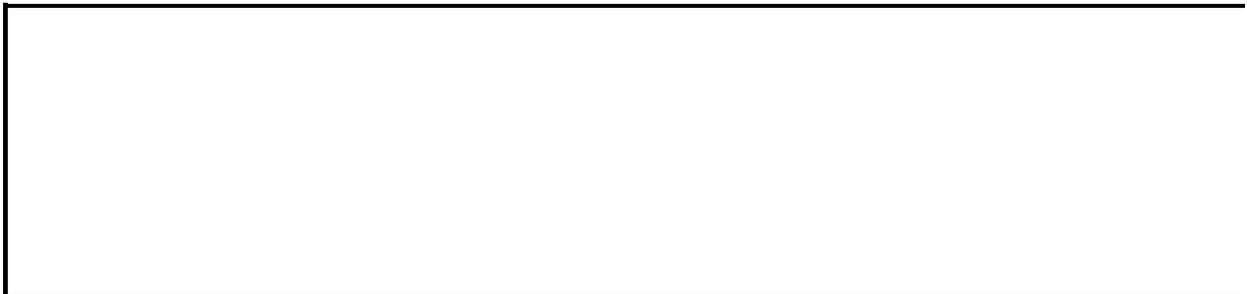
```
Addins("C:\Templates\FindAll.wll").Installed = True
```

This example loads Custom.dot.

```
AddIns("C:\Program Files\Microsoft Office\" _  
    & "Templates\Custom.dot").Installed = True
```

This example displays a message on the status bar if Dot1.dot is loaded as a global template.

```
If AddIns("Dot1.dot").Installed = True Then _  
    StatusBar = "Dot1.dot is loaded"
```



↳ [Show All](#)

International Property

Returns information about the current country/region and international settings.
Read-only **Variant**.

expression.**International**(*Index*)

expression Required. An expression that returns an **Application** object.

Index Required [WdInternationalIndex](#). The current country/region and/or international setting.

WdInternationalIndex can be one of these WdInternationalIndex constants.

wd24HourClock Returns **True** if you're using 24-hour time; returns **False** if you're using 12-hour time.

wdCurrencyCode Returns the currency symbol (\$ in U.S. English).

wdDateSeparator Returns the date separator (/ in U.S. English).

wdDecimalSeparator Returns the decimal separator (. in U.S. English).

wdInternationalAM Returns the string used to indicate morning hours (for example, 10 AM).

wdInternationalPM Returns the string used to indicate afternoon and evening hours (for example, 2 PM).

wdListSeparator Returns the list separator (, in U.S. English).

wdProductLanguageID Returns the language version of Word.

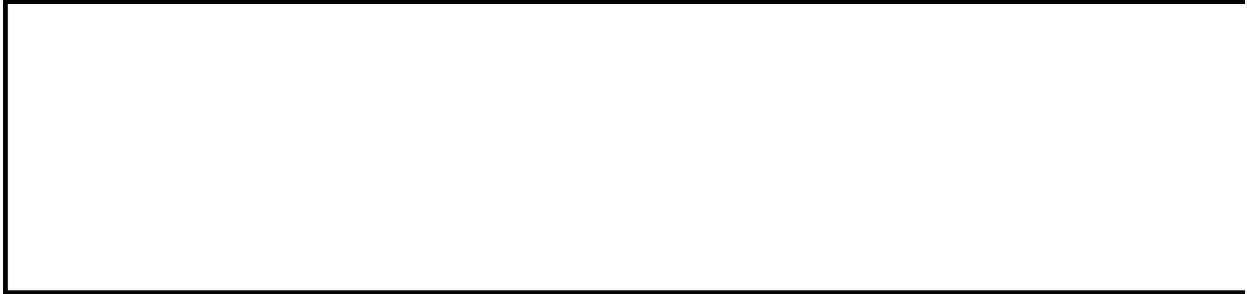
wdThousandsSeparator Returns the thousands separator (, in U.S. English).

wdTimeSeparator Returns the time separator (: in U.S. English).

Example

This example displays the currency format in the status bar.

```
StatusBar = "Currency Format: " _  
    & Application.International(wdCurrencyCode)
```



↳ [Show All](#)

InterpretHighAnsi Property

Returns or sets the high-ANSI text interpretation behavior. Read/write [WdHighAnsiText](#).

WdHighAnsiText can be one of these WdHighAnsiText constants.

wdAutoDetectHighAnsiFarEast Microsoft Word interprets high-ANSI text as East Asian characters only if Word automatically detects East Asian language text.

wdHighAnsiIsHighAnsi Word interprets all high-ANSI text as East Asian characters.

wdHighAnsiIsFarEast Word doesn't interpret any high-ANSI text as East Asian characters.

expression.**InterpretHighAnsi**

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

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Word to interpret all high-ANSI text as East Asian characters.

```
Options.InterpretHighAnsi = wdHighAnsiIsFarEast
```



InUse Property

-
True if the specified style is a built-in style that has been modified or applied in the document or a new style that has been created in the document. Read-only **Boolean**.

expression.**InUse**

expression Required. An expression that returns a [Style](#) object.

Remarks

This property doesn't necessarily indicate whether the style is currently applied to any text in the document. For instance, if text that's been formatted with a style is deleted, the **InUse** property of the style remains **True**. For built-in styles that have never been used in the document, this property returns **False**.

Example

This example displays a message box that lists the names of all the styles that are currently being used in the active document.

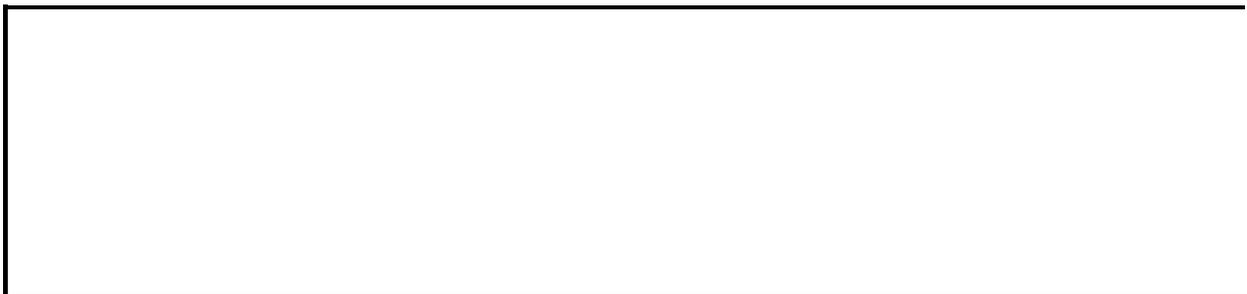
```
Dim docActive As Document
Dim strMessage As String
Dim styleLoop As Style

Set docActive = ActiveDocument

strMessage = "Styles in use:" & vbCrLf

For Each styleLoop In docActive.Styles
    If styleLoop.InUse = True Then
        With docActive.Content.Find
            .ClearFormatting
            .Text = ""
            .Style = styleLoop
            .Execute Format:=True
            If .Found = True Then
                strMessage = strMessage & styleLoop.Name & vbCrLf
            End If
        End With
    End If
Next styleLoop

MsgBox strMessage
```



InvalidAddress Property

True for Microsoft Word to mark a record in a mail merge data source if it contains invalid data in an address field. Read/write **Boolean**.

expression.**InvalidAddress**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

Remarks

Use the [SetAllErrorFlags](#) method to set both the **InvalidAddress** and **InvalidComments** properties for all records in a data source.

Example

This example loops through the records in the mail merge data source and checks whether the ZIP code field (in this case field number six) contains less than five digits. If a record does contain a ZIP code of less than five digits, the record is excluded from the mail merge and the address is marked as invalid.

```
Sub ExcludeRecords()

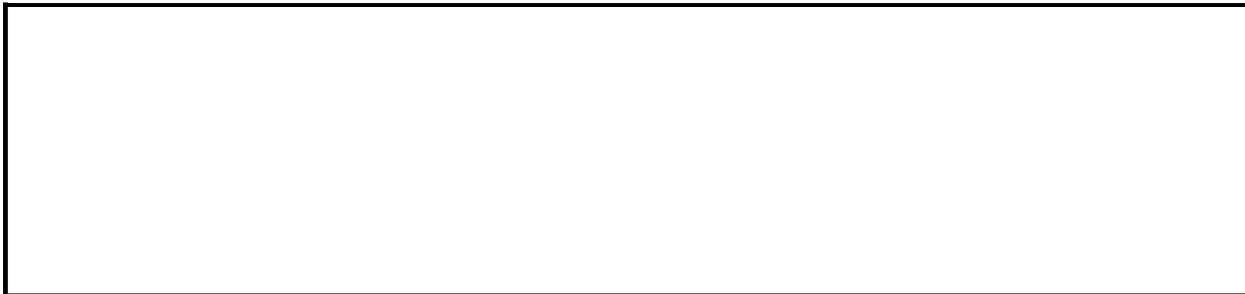
    Dim intCount As Integer

    On Error Resume Next

    With ActiveDocument.MailMerge.DataSource
        .ActiveRecord = wdFirstRecord
        Do
            intCount = intCount + 1
            'Counts the number of digits in the postal code field an
            'it is less than 5, the record is excluded from the mail
            'marked as having an invalid address, and given a commen
            'describing why the postal code was removed
            If Len(.DataFields(6).Value) < 5 Then
                .Included = False
                .InvalidAddress = True
                .InvalidComments = "The zip code for this record" &
                    "is less than five digits. This record is" & _
                    "removed from the mail merge process."
            End If

            .ActiveRecord = wdNextRecord
        Loop Until intCount >= .ActiveRecord
    End With

End Sub
```



InvalidComments Property

-
If the [InvalidAddress](#) property is **True**, returns or sets a **String** that describes an invalid address error. 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 loops through the records in the mail merge data source and checks whether the ZIP code field (in this case field number six) contains less than five digits. If a record does contain a ZIP code of less than five digits, the record is excluded from the mail merge, the address is marked as invalid, and a comment why the record was excluded.

```
Sub ExcludeRecords()  
  
    Dim intCount As Integer  
  
    On Error Resume Next  
  
    With ActiveDocument.MailMerge.DataSource  
        .ActiveRecord = wdFirstRecord  
        Do  
            intCount = intCount + 1  
            'Counts the number of digits in the postal code field and  
            'it is less than 5, the record is excluded from the mail  
            'marked as having an invalid address, and given a comment  
            'describing why the postal code was removed  
            If Len(.DataFields(6).Value) < 5 Then  
                .Included = False  
                .InvalidAddress = True  
                .InvalidComments = "The zip code for this record" &  
                    "is less than five digits. This record is" & _  
                    "removed from the mail merge process."  
            End If  
  
            .ActiveRecord = wdNextRecord  
        Loop Until intCount >= .ActiveRecord  
    End With  
  
End Sub
```



IPAtEndOfLine Property

-

True if the insertion point is at the end of a line that wraps to the next line. **False** if the selection isn't collapsed, if the insertion point isn't at the end of a line, or if the insertion point is positioned before a paragraph mark. Read-only **Boolean**.

Example

If the insertion point isn't already at the end of the line, this example moves it there.

```
Selection.Collapse Direction:=wdCollapseEnd
If Selection.IPAtEndOfLine = False Then
    Selection.EndKey Unit:=wdLine, Extend:=wdMove
End If
```



IsEndOfRowMark Property

-
True if the specified selection or range is collapsed and is located at the end-of-row mark in a table. Read-only **Boolean**.

Note This property is the equivalent of the following expression:

```
Selection.Information(wdAtEndOfRowMarker)
```

Example

This example collapses the selection and selects the current row if the insertion point is at the end of the row (just before the end-of-row mark).

```
Selection.Collapse Direction:=wdCollapseEnd
If Selection.IsEndOfRowMark = True Then
    Selection.Rows(1).Select
End If
```



IsFirst Property

-
True if the specified column or row is the first one in the table. Read-only **Boolean**.

expression.**IsFirst**

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

Example

This example determines whether the first row in the selection is the first row in the table.

```
MsgBox Selection.Rows(1).IsFirst
```



IsHeader Property

True if the specified [HeaderFooter](#) object is a header. Read-only **Boolean**.

expression.**IsHeader**

expression Required. An expression that returns a [HeaderFooter](#) object.

Example

This example selects the footer and adds a page number.

```
With ActiveDocument.ActiveWindow.ActivePane.View  
    .Type = wdPrintView  
    .SeekView = wdSeekCurrentPageHeader  
End With
```

```
If Selection.HeaderFooter.IsHeader = True Then  
    ActiveDocument.ActiveWindow.ActivePane.View _  
        .SeekView = wdSeekCurrentPageFooter  
End If
```

```
Selection.HeaderFooter.PageNumbers.Add
```



IsLast Property

-
True if the specified column or row is the last one in the table. Read-only **Boolean**.

expression.**IsLast**

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

Example

This example determines whether the second row is the last row in the table.

```
MsgBox ActiveDocument.Tables(1).Rows(2).IsLast
```

This example determines whether the first column in the selection is the last column in the table.

```
If Selection.Information(wdWithInTable) = True Then  
    MsgBox Selection.Columns(1).IsLast  
End If
```



IsMasterDocument Property

-
True if the specified document is a master document. A master document includes one or more subdocuments. Read-only **Boolean**.

Example

If the active document is a master document, this example switches to master document view and opens the first subdocument.

```
If ActiveDocument.IsMasterDocument = True Then
    ActiveDocument.ActiveWindow.View.Type = wdMasterView
    ActiveDocument.Subdocuments(1).Open
Else
    MsgBox "This document is not a master document."
End If
```



IsValid Property

True if the specified variable that references an object is valid. **False** if the object referenced by the variable has been deleted. Read-only **Boolean**.

expression.IsValid(*Object*)

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

Object Required **Object**. A variable that references an object.

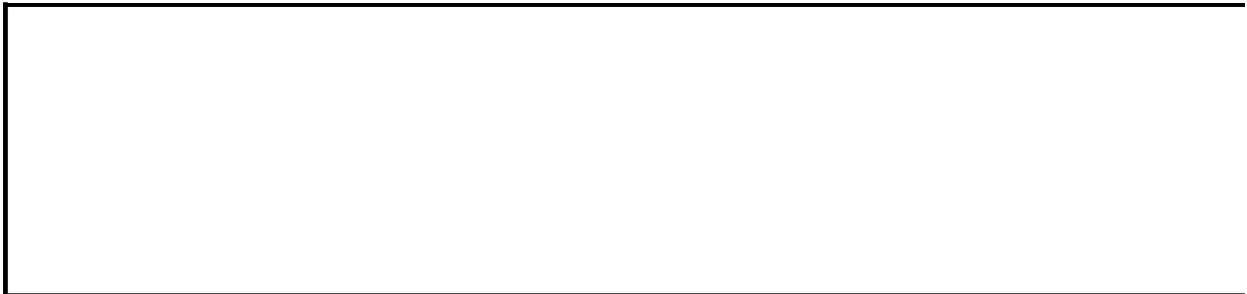
Example

This example adds a table to the active document and assigns it to the variable `aTable`. The example then deletes the first table from the document. If the table that `aTable` refers to was not the first table in the document (that is, if `aTable` is still a valid object), the example also removes any borders from that table.

```
Dim aTable As Table
```

```
Set aTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _  
    NumRows:=2, NumColumns:=3)
```

```
ActiveDocument.Tables(1).Delete  
If IsObjectValid(aTable) = True Then _  
    aTable.Borders.Enable = False
```



IsPictureBullet Property

True indicates that an [InlineShape](#) object is a picture bullet. Read-only **Boolean**.

expression.**IsPictureBullet**

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

Remarks

Although picture bullets are considered inline shapes, searching a document's [InlineShapes](#) collection will not return picture bullets.

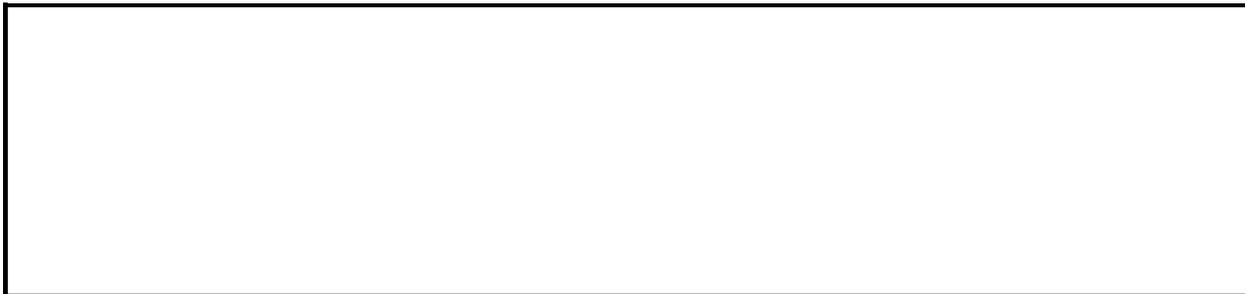
Example

This example formats the selected list if the list is formatted with a picture bullet. If not, a message is displayed.

```
Sub IsSelectionAPictureBullet(shp As InlineShape)
    On Error GoTo ErrorHandler
    If shp.IsPictureBullet = True Then
        shp.Width = InchesToPoints(0.5)
        shp.Height = InchesToPoints(0.05)
    End If
    Exit Sub
ErrorHandler:
    MsgBox "The selection is not a list or " & _
        "does not contain picture bullets."
End Sub
```

Use the following code to call the routine above.

```
Sub CallPic()
    Call IsSelectionAPictureBullet(shp:=Selection _
        .Range.ListFormat.ListPictureBullet)
End Sub
```



IsStyleSeparator Property

True if a paragraph contains a special hidden paragraph mark that allows Microsoft Word to appear to join paragraphs of different paragraph styles. Read-only **Boolean**.

expression.**IsStyleSeparator**

expression Required. An expression that returns a [Paragraph](#) object.

Example

This example formats all paragraphs in which there is a style separator with the built-in "Normal" style.

```
Sub StyleSep()  
    Dim pghDoc As Paragraph  
    For Each pghDoc In ThisDocument.Paragraphs  
        If pghDoc.IsStyleSeparator = True Then  
            pghDoc.Range.Select  
            Selection.Style = "Normal"  
        End If  
    Next pghDoc  
End Sub
```

This example adds a paragraph after each style separator and then deletes the style separator.

```
Sub RemoveStyleSeparator()  
    Dim pghDoc As Paragraph  
    Dim styName As String  
  
    'Loop through all paragraphs in document to check if it is a sty  
    'separator. If it is, delete it and enter a regular paragraph  
    For Each pghDoc In ThisDocument.Paragraphs  
        If pghDoc.IsStyleSeparator = True Then  
            pghDoc.Range.Select  
            With Selection  
                .Collapse (wdCollapseEnd)  
                .TypeParagraph  
                .MoveLeft (1)  
                .TypeBackspace  
            End With  
        End If  
    Next pghDoc  
End Sub
```



IsSubdocument Property

-

True if the specified document is opened in a separate document window as a subdocument of a master document. Read-only **Boolean**

Example

This example determines whether Sales.doc is a subdocument and then displays a message indicating its status.

```
If Documents("Sales.doc").IsSubdocument = True Then
    MsgBox "Sales.doc is a subdocument."
Else
    MsgBox "Sales.doc is not a subdocument."
End If
```



Italic Property

-
True if the font or range is formatted as italic. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

Example

This example formats the first word in the active document as italic.

```
ActiveDocument.Words(1).Italic = True
```

This example checks the selection for italic formatting and removes any that it finds.

```
If Selection.Type = wdSelectionNormal Then
    mySel = Selection.Font.Italic
    If mySel = wdUndefined or mySel = True Then
        MsgBox "There's italic text in selection. " _
            & "Click OK to remove."
        Selection.Font.Italic = False
    Else
        MsgBox "No italic text in the selection."
    End If
Else
    MsgBox "You need to select some text."
End If
```



ItalicBi Property

-
True if the font or range is formatted as italic. Returns **True**, **False** or **wdUndefined** (for a mixture of italic and non-italic text). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**ItalicBi**

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

Remarks

The **ItalicBi** property applies to text in right-to-left languages. For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example italicizes the first paragraph in the active right-to-left language document.

```
ActiveDocument.Paragraphs(1).Range.ItalicBi = True
```



Item Property

Returns or sets the adjustment value specified by the **Index** argument. 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. Read/write **Single**.

expression.**Item**(**Index**)

expression Required. An expression that returns an [Adjustments](#) object.

Index Required **Long**. The index number of the adjustment.

Remarks

AutoShapes and WordArt objects have up to eight adjustments.

Example

This example adds two crosses to the active document and then sets the value for adjustment one (the only one for this type of AutoShape) on each cross.

```
Dim docActive As Document

Set docActive = ActiveDocument

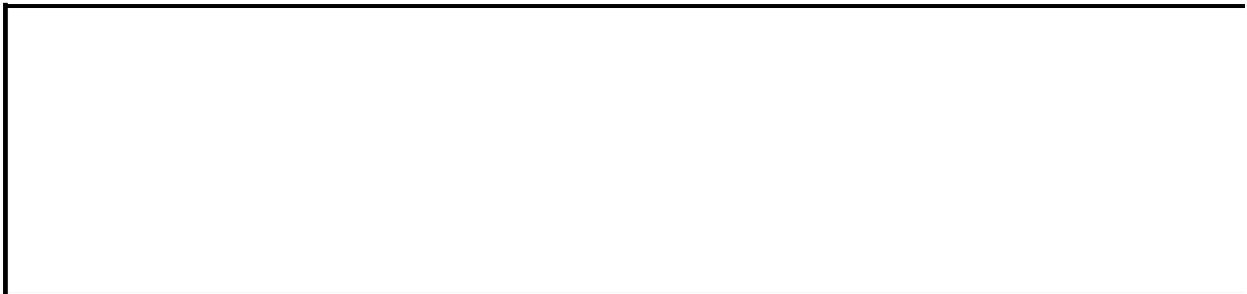
With docActive.Shapes
    .AddShape(msoShapeCross, _
        10, 10, 100, 100).Adjustments.Item(1) = 0.4
    .AddShape(msoShapeCross, _
        150, 10, 100, 100).Adjustments.Item(1) = 0.2
End With
```

This example has the same result as the previous example even though it doesn't explicitly use the **Item** property.

```
Dim docActive As Document

Set docActive = ActiveDocument

With docActive.Shapes
    .AddShape(msoShapeCross, _
        10, 10, 100, 100).Adjustments(1) = 0.4
    .AddShape(msoShapeCross, _
        150, 10, 100, 100).Adjustments(1) = 0.2
End With
```



JoinBorders Property

True if vertical borders at the edges of paragraphs and tables are removed so that the horizontal borders can connect to the page border. Read/write **Boolean**.

expression.**JoinBorders**

expression Required. An expression that returns a [Borders](#) object.

Example

This example adds a border around each page in the first section of the selection. The example also removes the horizontal borders at the edges of tables and paragraphs, thus connecting the horizontal borders to the page border.

```
Dim borderLoop As Border

With Selection.Sections(1)
    For Each borderLoop In .Borders
        borderLoop.ArtStyle = wdArtBalloonsHotAir
        borderLoop.ArtWidth = 15
    Next borderLoop
    With .Borders
        .DistanceFromLeft = 2
        .DistanceFromRight = 2
        .DistanceFrom = wdBorderDistanceFromText
        .JoinBorders = True
    End With
End With
```



JustificationMode Property

Returns or sets the character spacing adjustment for the specified document.
Read/write [WdJustificationMode](#).

WdJustificationMode can be one of these WdJustificationMode constants.

wdJustificationModeCompress

wdJustificationModeCompressKana

wdJustificationModeExpand

expression.**JustificationMode**

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

Example

This example sets Microsoft Word to compress only punctuation marks when adjusting character spacing.

```
ActiveDocument.JustificationMode = wdJustificationModeCompressKana
```



↳ [Show All](#)

Kana Property

Returns or sets whether the specified range of Japanese language text is hiragana or katakana. Read/write [WdKana](#).

WdKana can be one of these WdKana constants.

wdKanaHiragana

wdKanaKatakana

expression.**Kana**

expression Required. An expression that returns a [Range](#) object.

Remarks

This property returns **wdUndefined** if the range contains a mix of hiragana and katakana or if the range contains some non-Japanese text. If you set the **Kana** property to **wdUndefined**, an error occurs.

Example

This example displays what type of Japanese text the current selection contains.

```
Select Case Selection.Range.Kana
  Case wdKanaHiragana
    MsgBox "This text is hiragana."
  Case wdKanaKatakana
    MsgBox "This text is katakana."
  Case wdUndefined
    MsgBox "This text is a mix of " _
      & "hiragana and katakana."
End Select
```



KeepEntryFormatting Property

True if formatting from table of authorities entries is applied to the entries in the specified table of authorities. Corresponds to the \f switch for a Table of Authorities (TOA) field. Read/write **Boolean**.

expression.**KeepEntryFormatting**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example removes the formatting from the entries in the first table of authorities of the active document (the \f switch is added to the TOA field).

```
If ActiveDocument.TablesOfAuthorities.Count >= 1 Then
    ActiveDocument.TablesOfAuthorities(1) _
        .KeepEntryFormatting = False
End If
```



KeepTogether Property

-

True if all lines in the specified paragraphs remain on the same page when Microsoft Word repaginates the document. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

Example

This example formats the paragraphs in the active document so that all the lines in each paragraph are on the same page when Word repaginates the document.

```
ActiveDocument.Paragraphs.KeepTogether = True
```



KeepWithNext Property

-

True if the specified paragraph remains on the same page as the paragraph that follows it when Microsoft Word repaginates the document. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

Example

This example keeps the third paragraph through sixth paragraph in the active document on the same page.

```
For i = 3 To 5  
    ActiveDocument.Paragraphs(i).KeepWithNext = True  
Next i
```



↳ [Show All](#)

KernedPairs Property

-
Indicates that character pairs in a WordArt object have been kerned. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

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 document.

```
Sub Kerned()  
    With ActiveDocument.Range(1, ActiveDocument.Shapes.Count).ShapeR  
        If .Type = msoTextEffect Then  
            .TextEffect.KernedPairs = True  
        End If  
    End With  
End Sub
```



Kerning Property

Returns or sets the minimum font size for which Microsoft Word will adjust kerning automatically. Read/write **Single**.

expression.**Kerning**

expression Required. An expression that returns a [Font](#) object.

Example

This example sets the minimum font size for automatic kerning to 12 points or larger in the active document.

```
ActiveDocument.Content.Font.Kerning = 12
```

This example displays the minimum font size for which Word will automatically adjust kerning in the selected text.

```
If Selection.Type = wdSelectionNormal Then  
    MsgBox Selection.Font.Kerning  
Else  
    MsgBox "You need to select some text."  
End If
```



KerningByAlgorithm Property

-
True if Microsoft Word kerns half-width Latin characters and punctuation marks in the specified document. Read/write **Boolean**.

Example

This example sets Microsoft Word to kern half-width Latin characters and punctuation marks in the active document.

```
ActiveDocument.KerningByAlgorithm = True
```



KeyBindings Property

Returns a [KeyBindings](#) collection that represents customized key assignments, which include a key code, a key category, and a command.

expression.**KeyBindings**

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

Example

This example assigns the CTRL+ALT+W key combination to the FileClose command. This keyboard customization is saved in the Normal template.

```
CustomizationContext = NormalTemplate
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, _
    wdKeyW), KeyCategory:=wdKeyCategoryCommand, _
    Command:="FileClose"
```

This example inserts the command name and key combination string for each item in the [KeyBindings](#) collection.

```
Dim kbLoop As KeyBinding

CustomizationContext = NormalTemplate
For Each kbLoop In KeyBindings
    Selection.InsertAfter kbLoop.Command & vbTab _
        & kbLoop.KeyString & vbCr
    Selection.Collapse Direction:=wdCollapseEnd
Next kbLoop
```



↳ [Show All](#)

KeyCategory Property

Returns the type of item assigned to the specified key binding. Read-only [WdKeyCategory](#).

WdKeyCategory can be one of these WdKeyCategory constants.

wdKeyCategoryAutoText

wdKeyCategoryCommand

wdKeyCategoryDisable

wdKeyCategoryFont

wdKeyCategoryMacro

wdKeyCategoryNil

wdKeyCategoryPrefix

wdKeyCategoryStyle

wdKeyCategorySymbol

expression.**KeyCategory**

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

Example

This example displays the keys assigned to font names. A message is displayed if no keys have been assigned to fonts.

```
Dim kbLoop As KeyBinding
Dim intCount As Integer

intCount = 0

For Each kbLoop In KeyBindings
    If kbLoop.KeyCategory = wdKeyCategoryFont Then
        intCount = intCount + 1
        MsgBox kbLoop.Command & vbCr & kbLoop.KeyString
    End If
Next kbLoop

If intCount = 0 Then _
    MsgBox "Keys haven't been assigned to fonts"
```



KeyCode Property

Returns a unique number for the first key in the specified key binding. Read-only **Long**.

Note You create this number by using the [BuildKeyCode](#) method when you're adding key bindings by using the [Add](#) method of the [KeyBindings](#) object.

expression.**KeyCode**

expression Required. An expression that returns a [KeyBinding](#) object.

Example

This example displays a message if the **KeyBindings** collection includes the ALT+CTRL+W key combination.

```
Dim lngCode As Long
Dim kbLoop As KeyBinding

CustomizationContext = NormalTemplate
lngCode = BuildKeyCode(wdKeyAlt, wdKeyControl, wdKeyW)
For Each kbLoop In KeyBindings
    If lngCode = kbLoop.KeyCode Then
        MsgBox kbLoop.KeyString & " is already in use"
    End If
Next kbLoop
```



KeyCode2 Property

Returns a unique number for the second key in the specified key binding. Read-only **Long**.

expression.**KeyCode2**

expression Required. An expression that returns a [KeyBinding](#) object.

Example

This example displays the key codes of each key in the [KeyBindings](#) collection (the collection of all the customized keys in the active document).

```
Dim aKey As KeyBinding

CustomizationContext = ActiveDocument
For Each aKey In KeyBindings
    If aKey.KeyCode2 <> wdNoKey Then
        MsgBox aKey.KeyString & vbCr _
            & "KeyCode1 = " & aKey.KeyCode & vbCr _
            & "KeyCode2 = " & aKey.KeyCode2
    Else
        MsgBox aKey.KeyString & vbCr _
            & "KeyCode1 = " & aKey.KeyCode
    End If
Next aKey
```



↳ [Show All](#)

KeysBoundTo Property

Returns a [KeysBoundTo](#) object that represents all the key combinations assigned to the specified item.

expression.**KeysBoundTo**(*KeyCategory*, *Command*, *CommandParameter*)

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

KeyCategory Required [WdKeyCategory](#). The category of the key combination.

WdKeyCategory can be one of these WdKeyCategory constants.

wdKeyCategoryAutoText

wdKeyCategoryCommand

wdKeyCategoryDisable

wdKeyCategoryFont

wdKeyCategoryMacro

wdKeyCategoryNil

wdKeyCategoryPrefix

wdKeyCategoryStyle

wdKeyCategorySymbol

Command Required **String**. The name of the command.

CommandParameter Optional **Variant**. Additional text, if any, required for the command specified by **Command**. For more information, see the "Remarks" section in the [Add](#) method for the [KeyBindings](#) object.

Example

This example displays all the key combinations assigned to the FileOpen command in the template attached to the active document.

```
Dim kbLoop As KeyBinding
Dim strOutput As String

CustomizationContext = ActiveDocument.AttachedTemplate

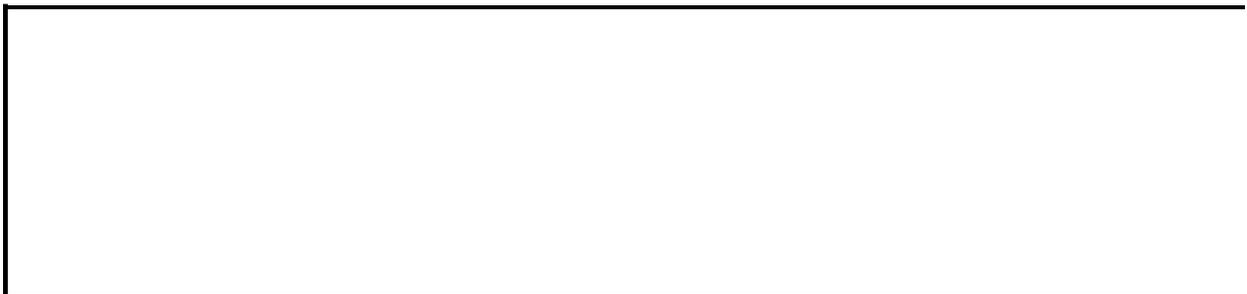
For Each kbLoop In _
    KeysBoundTo(KeyCategory:=wdKeyCategoryCommand, _
        Command:="FileOpen")
    strOutput = strOutput & kbLoop.KeyString & vbCr
Next kbLoop

MsgBox strOutput
```

This example removes all key assignments from Macro1 in the Normal template.

```
Dim aKey As KeyBinding

CustomizationContext = NormalTemplate
For Each aKey In _
    KeysBoundTo(KeyCategory:=wdKeyCategoryMacro, _
        Command:="Macro1")
    aKey.Disable
Next aKey
```



KeyString Property

Returns the key combination string for the specified keys (for example, CTRL+SHIFT+A). Read-only **String**.

expression.**KeyString**

expression Required. An expression that returns a [KeyBinding](#) object.

Example

This example displays the key combination string for the first customized key combination in the Normal template.

```
CustomizationContext = NormalTemplate
If KeyBindings.Count >= 1 Then
    MsgBox KeyBindings(1).KeyString
End If
```

This example displays a message if the [KeyBindings](#) collection includes the ALT+CTRL+W key combination.

```
Dim aCode As Long
Dim aKey As KeyBinding

CustomizationContext = NormalTemplate
aCode = BuildKeyCode(wdKeyAlt, wdKeyControl, wdKeyW)
For Each aKey In KeyBindings
    If aCode = aKey.KeyCode Then
        MsgBox aKey.KeyString & " is already in use"
    End If
Next aKey
```



↳ [Show All](#)

Kind Property

▶ [Kind property as it applies to the **Document** object.](#)

Returns or sets the format type that Microsoft Word uses when automatically formatting the specified document. Read/write [WdDocumentKind](#).

WdDocumentKind can be one of these WdDocumentKind constants.

wdDocumentEmail

wdDocumentNotSpecified

wdDocumentLetter

expression.**Kind**

expression Required. An expression that returns a [Document](#) object.

▶ [Kind property as it applies to the **Field** object.](#)

Returns the type of link for a [Field](#) object. Read-only [WdFieldKind](#).

WdFieldKind can be one of these WdFieldKind constants.

wdFieldKindCold A field that doesn't have a result, for example, an Index Entry (XE), Table of Contents Entry (TC), or Private field.

wdFieldKindHot A field that's automatically updated each time it's displayed or each time the page is reformatted, but which can also be manually updated (for example, INCLUDEPICTURE or FORMDROPDOWN).

wdFieldKindNone An invalid field (for example, a pair of field characters with nothing inside).

wdFieldKindWarm A field that can be updated and has a result. This type includes fields that are automatically updated when the source changes as well as fields that can be manually updated (for example, DATE or INCLUDETTEXT).

expression.**Kind**

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

Example

▶ [As it applies to the **Document** object.](#)

This example asks the user whether the active document is an e-mail message. If the response is Yes, the document is formatted as an e-mail message.

```
response = MsgBox("Is this document an email message?", vbYesNo)
If response = vbYes Then
    ActiveDocument.Kind = wdDocumentEmail
    ActiveDocument.Content.AutoFormat
End If
```

▶ [As it applies to the **Field** object.](#)

This example updates all warm link fields in the active document.

```
For Each aField In ActiveDocument.Fields
    If aField.Kind = wdFieldKindWarm Then aField.Update
Next aField
```



Label Property

Returns a string that's used to identify the portion of the source file that's being linked. For example, if the source file is a Microsoft Excel workbook, the **Label** property might return "Workbook1!R3C1:R4C2" if the OLE object contains only a few cells from the worksheet. Read-only **String**.

Note This property works only for shapes, inline shapes, or fields that are linked OLE objects.

expression.**Label**

expression Required. An expression that returns an [OLEFormat](#) object.

Example

This example returns the label for the first field in the active document.

```
MsgBox ActiveDocument.Fields(1).OLEFormat.Label
```



LabelSmartTags Property

True for Microsoft Word to mark text in documents with smart tag information.
Read/write **Boolean**.

expression.**LabelSmartTags**

expression Required. An expression that returns an [Options](#) object.

Example

This example turns off marking smart tags in documents.

```
Sub MarkSmartTags()  
    Application.Options.LabelSmartTags = False  
End Sub
```



LandscapeFontNames Property

Returns a [FontNames](#) object that includes the names of all the available landscape fonts.

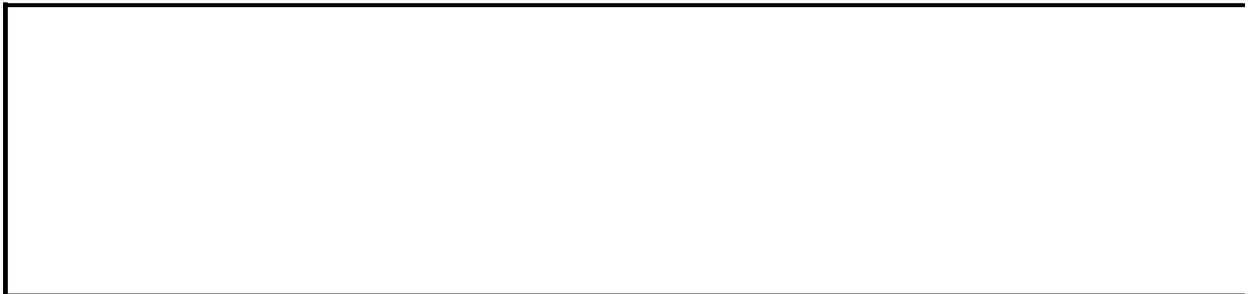
expression.**LandscapeFontNames**

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

Example

This example creates a sorted list in a new document of the landscape font names in the **FontNames** object.

```
Sub ListLandscapeFonts()  
    Dim docNew As Document  
    Dim intCount As Integer  
  
    Set docNew = Documents.Add  
    docNew.Content.InsertAfter "Landscape Fonts" & vbCrLf  
  
    For intCount = 1 To LandscapeFontNames.Count  
        docNew.Content.InsertAfter LandscapeFontNames(intCount) _  
            & vbCrLf  
    Next  
  
    With docNew  
        .Range(Start:=.Paragraphs(2).Range.Start, End:=.Paragraphs _  
            (docNew.Paragraphs.Count).Range.End).Select  
    End With  
  
    Selection.Sort  
End Sub
```



↳ [Show All](#)

Language Property

Returns an [MsoLanguageID](#) constant that represents the language selected for the Microsoft Word user interface.

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

expression.**Language**

expression Required. An expression that returns an [Application](#) object.

Remarks

The value of this property is the same as the value returned by the following expression:

```
Application.LanguageSettings _  
    .LanguageID(msoLanguageIDUI)
```

Example

This example displays a message stating whether the language selected for the Microsoft Word 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
```



LanguageDesignation Property

Returns the designated language of the system software. Read-only **String**.

expression.**LanguageDesignation**

expression Required. An expression that returns a [System](#) object.

Example

This example displays "U.S. English" if the **LanguageDesignation** property returns "English (US)".

```
If System.LanguageDesignation = "English (US)" Then _  
    MsgBox "U.S. English"
```



LanguageDetected Property

Returns or sets a value that specifies whether Microsoft Word has detected the language of the specified text. Read/write **Boolean**.

expression.**LanguageDetected**

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

Remarks

Check the [LanguageID](#) property for the results of any previous language detection.

The **LanguageDetected** property is set to **True** when the [DetectLanguage](#) method is called. To reevaluate the language of the specified text, you must first set the **LanguageDetected** property to **False**.

For more information about automatic language detection, see [About automatic language detection](#).

Example

This example checks the active document to determine the language it's written in and then displays the result.

With ActiveDocument

```
    If .LanguageDetected = True Then
        x = MsgBox("This document has already " _
            & "been checked. Do you want to check " _
            & "it again?", vbYesNo)
        If x = vbYes Then
            .LanguageDetected = False
            .DetectLanguage
        End If
    Else
        .DetectLanguage
    End If
    If .Range.LanguageID = wdEnglishUS Then
        MsgBox "This is a U.S. English document."
    Else
        MsgBox "This is not a U.S. English document."
    End If
```

End With



↳ [Show All](#)

LanguageID Property

Returns or sets the language for the specified object. Read/write [WdLanguageID](#).

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdAmharic

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt

wdArabicIraq

wdArabicJordan

wdArabicKuwait

wdArabicLebanon

wdArabicLibya

wdArabicMorocco

wdArabicOman

wdArabicQatar

wdArabicSyria

wdArabicTunisia

wdArabicUAE

wdArabicYemen

wdArmenian

wdAssamese

wdAzeriCyrillic

wdAzeriLatin

wdBasque

wdBelgianDutch
wdBelgianFrench
wdBengali
wdBrazilianPortuguese
wdBulgarian
wdBurmese
wdByelorussian
wdCatalan
wdCherokee
wdChineseHongKong
wdChineseMacao
wdChineseSingapore
wdCroatian
wdCzech
wdDanish
wdDivehi
wdDutch
wdEdo
wdEnglishAUS
wdEnglishBelize
wdEnglishCanadian
wdEnglishCaribbean
wdEnglishIreland
wdEnglishJamaica
wdEnglishNewZealand
wdEnglishPhilippines
wdEnglishSouthAfrica
wdEnglishTrinidad
wdEnglishUK
wdEnglishUS
wdEnglishZimbabwe
wdEstonian
wdFaeroese

wdFarsi
wdFilipino
wdFinnish
wdFrench
wdFrenchCameroon
wdFrenchCanadian
wdFrenchCotedIvoire
wdFrenchLuxembourg
wdFrenchMali
wdFrenchMonaco
wdFrenchReunion
wdFrenchSenegal
wdFrenchWestIndies
wdFrenchZaire
wdFrisianNetherlands
wdFulfulde
wdGaelicIreland
wdGaelicScotland
wdGalician
wdGeorgian
wdGerman
wdGermanAustria
wdGermanLiechtenstein
wdGermanLuxembourg
wdGreek
wdGuarani
wdGujarati
wdHausa
wdHawaiian
wdHebrew
wdHindi
wdHungarian
wdIbibio

wdIcelandic
wdIgbo
wdIndonesian
wdInuktitut
wdItalian
wdJapanese
wdKannada
wdKanuri
wdKashmiri
wdKazakh
wdKhmer
wdKirghiz
wdKonkani
wdKorean
wdKyrgyz
wdLanguageNone
wdLao
wdLatin
wdLatvian
wdLithuanian
wdMacedonian
wdMalayalam
wdMalayBruneiDarussalam
wdMalaysian
wdMaltese
wdManipuri
wdMarathi
wdMexicanSpanish
wdMongolian
wdNepali
wdNoProofing
wdNorwegianBokmol
wdNorwegianNynorsk

wdOriya
wdOromo
wdPashto
wdPolish
wdPortuguese
wdPunjabi
wdRhaetoRomanic
wdRomanian
wdRomanianMoldova
wdRussian
wdRussianMoldova
wdSamiLappish
wdSanskrit
wdSerbianCyrillic
wdSerbianLatin
wdSesotho
wdSimplifiedChinese
wdSindhi
wdSindhiPakistan
wdSinhalese
wdSlovak
wdSlovenian
wdSomali
wdSorbian
wdSpanish
wdSpanishArgentina
wdSpanishBolivia
wdSpanishChile
wdSpanishColombia
wdSpanishCostaRica
wdSpanishDominicanRepublic
wdSpanishEcuador
wdSpanishElSalvador

wdSpanishGuatemala
wdSpanishHonduras
wdSpanishModernSort
wdSpanishNicaragua
wdSpanishPanama
wdSpanishParaguay
wdSpanishPeru
wdSpanishPuertoRico
wdSpanishUruguay
wdSpanishVenezuela
wdSutu
wdSwahili
wdSwedish
wdSwedishFinland
wdSwissFrench
wdSwissGerman
wdSwissItalian
wdSyriac
wdTajik
wdTamazight
wdTamazightLatin
wdTamil
wdTatar
wdTelugu
wdThai
wdTibetan
wdTigrignaEritrea
wdTigrignaEthiopic
wdTraditionalChinese
wdTsonga
wdTswana
wdTurkish
wdTurkmen

wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin
wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdYi
wdYiddish
wdYoruba
wdZulu

expression.**LanguageID**

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

Remarks

For a custom dictionary, you must first set the [LanguageSpecific](#) property to **True** before specifying the the **LanguageID** property. Custom dictionaries that are language specific only look at text formatted for that language.

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 formats the second paragraph in the active document as French and then adds a new custom dictionary that will be used on the French text.

```
ActiveDocument.Paragraphs(2).Range.LanguageID = wdFrench
Set myDictionary = CustomDictionaries.Add(FileName:="French.dic")
With myDictionary
    .LanguageSpecific = True
    .LanguageID = wdFrench
End With
```

This example redefines the Title style to use the Spanish proofing tools. The new style description is then displayed in a message box.

```
ActiveDocument.Styles("Title").LanguageID = wdSpanish
MsgBox ActiveDocument.Styles("Title").Description
```



LanguageIDFarEast Property

Returns or sets an East Asian language for the specified object. Read/write [WdLanguageID](#).

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdAmharic

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt

wdArabicIraq

wdArabicJordan

wdArabicKuwait

wdArabicLebanon

wdArabicLibya

wdArabicMorocco

wdArabicOman

wdArabicQatar

wdArabicSyria

wdArabicTunisia

wdArabicUAE

wdArabicYemen

wdArmenian

wdAssamese

wdAzeriCyrillic

wdAzeriLatin

wdBelgianDutch

wdBengali
wdBulgarian
wdByelorussian
wdCherokee
wdChineseMacao
wdCroatian
wdDanish
wdEnglishAUS
wdEnglishCanadian
wdEnglishIreland
wdEnglishNewZealand
wdEnglishSouthAfrica
wdEnglishUK
wdEnglishZimbabwe
wdFaeroese
wdFinnish
wdFrenchCameroon
wdFrenchCotedIvoire
wdFrenchMali
wdFrenchReunion
wdFrenchWestIndies
wdFrisianNetherlands
wdGaelicScotland
wdGeorgian
wdGermanAustria
wdGermanLuxembourg
wdGujarati
wdHindi
wdIcelandic
wdInuktitut
wdJapanese
wdKashmiri
wdKhmer

wdKonkani
wdLanguageNone
wdLatvian
wdMacedonian
wdMalayBruneiDarussalam
wdMaltese
wdMarathi
wdMongolian
wdNoProofing
wdNorwegianNynorsk
wdPolish
wdPunjabi
wdRomanian
wdRussian
wdSamiLappish
wdSerbianCyrillic
wdSesotho
wdSindhi
wdSlovenian
wdSpanish
wdSpanishBolivia
wdSpanishColombia
wdSpanishDominicanRepublic
wdSpanishElSalvador
wdSpanishHonduras
wdSpanishNicaragua
wdSpanishParaguay
wdSpanishPuertoRico
wdSpanishVenezuela
wdSwahili
wdSwedishFinland
wdSwissGerman
wdTajik

wdTatar
wdThai
wdTraditionalChinese
wdTswana
wdBasque
wdBelgianFrench
wdBrazilianPortuguese
wdBurmese
wdCatalan
wdChineseHongKong
wdChineseSingapore
wdCzech
wdDutch
wdEnglishBelize
wdEnglishCaribbean
wdEnglishJamaica
wdEnglishPhilippines
wdEnglishTrinidad
wdEnglishUS
wdEstonian
wdFarsi
wdFrench
wdFrenchCanadian
wdFrenchLuxembourg
wdFrenchMonaco
wdFrenchSenegal
wdFrenchZaire
wdGaelicIreland
wdGalician
wdGerman
wdGermanLiechtenstein
wdGreek
wdHebrew

wdHungarian
wdIndonesian
wdItalian
wdKannada
wdKazakh
wdKirghiz
wdKorean
wdLao
wdLithuanian
wdMalayalam
wdMalaysian
wdManipuri
wdMexicanSpanish
wdNepali
wdNorwegianBokmol
wdOriya
wdPortuguese
wdRhaetoRomanic
wdRomanianMoldova
wdRussianMoldova
wdSanskrit
wdSerbianLatin
wdSimplifiedChinese
wdSlovak
wdSorbian
wdSpanishArgentina
wdSpanishChile
wdSpanishCostaRica
wdSpanishEcuador
wdSpanishGuatemala
wdSpanishModernSort
wdSpanishPanama
wdSpanishPeru

wdSpanishUruguay
wdSutu
wdSwedish
wdSwissFrench
wdSwissItalian
wdTamil
wdTelugu
wdTibetan
wdTsonga
wdTurkish
wdTurkmen
wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin
wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdZulu

expression.**LanguageIDFarEast**

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

Remarks

This is the recommended way to return or set the language of East Asian text in a document created in an East Asian version of Microsoft Word.

Example

This example sets the language of the selection to Korean.

```
Selection.LanguageIDFarEast = wdKorean
```



LanguageIDOther Property

Returns or sets the language for the specified object. Read/write [WdLanguageID](#).

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdAmharic

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt

wdArabicIraq

wdArabicJordan

wdArabicKuwait

wdArabicLebanon

wdArabicLibya

wdArabicMorocco

wdArabicOman

wdArabicQatar

wdArabicSyria

wdArabicTunisia

wdArabicUAE

wdArabicYemen

wdArmenian

wdAssamese

wdAzeriCyrillic

wdAzeriLatin

wdBelgianDutch

wdBengali
wdBulgarian
wdByelorussian
wdCherokee
wdChineseMacao
wdCroatian
wdDanish
wdEnglishAUS
wdEnglishCanadian
wdEnglishIreland
wdEnglishNewZealand
wdEnglishSouthAfrica
wdEnglishUK
wdEnglishZimbabwe
wdFaeroese
wdFinnish
wdFrenchCameroon
wdFrenchCotedIvoire
wdFrenchMali
wdFrenchReunion
wdFrenchWestIndies
wdFrisianNetherlands
wdGaelicScotland
wdGeorgian
wdGermanAustria
wdGermanLuxembourg
wdGujarati
wdHindi
wdIcelandic
wdInuktitut
wdJapanese
wdKashmiri
wdKhmer

wdKonkani
wdLanguageNone
wdLatvian
wdMacedonian
wdMalayBruneiDarussalam
wdMaltese
wdMarathi
wdMongolian
wdNoProofing
wdNorwegianNynorsk
wdPolish
wdPunjabi
wdRomanian
wdRussian
wdSamiLappish
wdSerbianCyrillic
wdSesotho
wdSindhi
wdSlovenian
wdSpanish
wdSpanishBolivia
wdSpanishColombia
wdSpanishDominicanRepublic
wdSpanishElSalvador
wdSpanishHonduras
wdSpanishNicaragua
wdSpanishParaguay
wdSpanishPuertoRico
wdSpanishVenezuela
wdSwahili
wdSwedishFinland
wdSwissGerman
wdTajik

wdTatar
wdThai
wdTraditionalChinese
wdTswana
wdBasque
wdBelgianFrench
wdBrazilianPortuguese
wdBurmese
wdCatalan
wdChineseHongKong
wdChineseSingapore
wdCzech
wdDutch
wdEnglishBelize
wdEnglishCaribbean
wdEnglishJamaica
wdEnglishPhilippines
wdEnglishTrinidad
wdEnglishUS
wdEstonian
wdFarsi
wdFrench
wdFrenchCanadian
wdFrenchLuxembourg
wdFrenchMonaco
wdFrenchSenegal
wdFrenchZaire
wdGaelicIreland
wdGalician
wdGerman
wdGermanLiechtenstein
wdGreek
wdHebrew

wdHungarian
wdIndonesian
wdItalian
wdKannada
wdKazakh
wdKirghiz
wdKorean
wdLao
wdLithuanian
wdMalayalam
wdMalaysian
wdManipuri
wdMexicanSpanish
wdNepali
wdNorwegianBokmol
wdOriya
wdPortuguese
wdRhaetoRomanic
wdRomanianMoldova
wdRussianMoldova
wdSanskrit
wdSerbianLatin
wdSimplifiedChinese
wdSlovak
wdSorbian
wdSpanishArgentina
wdSpanishChile
wdSpanishCostaRica
wdSpanishEcuador
wdSpanishGuatemala
wdSpanishModernSort
wdSpanishPanama
wdSpanishPeru

wdSpanishUruguay
wdSutu
wdSwedish
wdSwissFrench
wdSwissItalian
wdTamil
wdTelugu
wdTibetan
wdTsonga
wdTurkish
wdTurkmen
wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin
wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdZulu

expression.**LanguageIDOther**

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

Remarks

This is the recommended way to return or set the language of Latin text in a document created in a right-to-left language version of Microsoft Word.

Example

This example sets the language of the selection to French.

```
Selection.LanguageIDOther = wdFrench
```



Languages Property

Returns a [Languages](#) collection that represents the proofing languages listed in the **Language** dialog box (on the **Tools** menu, click **Language**, and then click **Set Language**).

expression.**Languages**

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 returns the full path and file name of the active spelling dictionary.

```
Dim dicSpell As Dictionary

Set dicSpell = _
    Languages(Selection.LanguageID).ActiveSpellingDictionary

MsgBox dicSpell.Path & Application.PathSeparator & dicSpell.Name
```

This example uses the aLang() array to store the proofing language names.

```
Dim intCount As Integer
Dim langLoop As Language
Dim aLang(Languages.Count - 1) As String

intCount = 0
For Each langLoop In Languages
    aLang(intCount) = langLoop.NameLocal
    intCount = intCount + 1
Next langLoop
```



LanguageSettings Property

Returns a [LanguageSettings](#) object, which contains information about the language settings in Microsoft Word.

expression.**LanguageSettings**

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



LanguageSpecific Property

-
True if the custom dictionary is to be used only with text formatted for a specific language. Read/write **Boolean**.

expression.**LanguageSpecific**

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

Example

This example checks to see whether any custom dictionaries are language specific. If any of them are, the example removes them from the list of active custom dictionaries.

```
Dim dicLoop As Dictionary

For each dicLoop in CustomDictionaries
    If dicLoop.LanguageSpecific = True Then dicLoop.Delete
Next dicLoop
```

This example adds a custom dictionary that will check only text that's formatted as German.

```
Dim dicNew As Dictionary

Set dicNew = CustomDictionaries.Add("German1.dic")
dicNew.LanguageSpecific = True
dicNew.LanguageID = wdGerman
```



↳ [Show All](#)

Last Property

▶ [Last property as it applies to the **Columns** object.](#)

Returns the last item in the **Columns** collection as a **Column** object.

expression.**Last**

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

▶ [Last property as it applies to the **Paragraphs** object.](#)

Returns the last item in the **Paragraphs** collection as a **Paragraph** object.

expression.**Last**

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

▶ [Last property as it applies to the **Characters, Sentences, and Words** objects.](#)

Returns a **Range** object that represents the last character, word, or sentence in a document, selection, or range.

expression.**Last**

expression Required. An expression that returns one of the above objects.

▶ [Last property as it applies to the **Rows** object.](#)

Returns the last item in the **Rows** collection as a **Row** object.

expression.**Last**

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

▶ [Last property as it applies to the **Sections** object.](#)

Returns the last item in the [Sections](#) collection as a [Section](#) object.

expression.**Last**

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

Example

▶ [As it applies to the **Paragraphs** object.](#)

This example formats the last paragraph in the active document to be right-aligned.

```
ActiveDocument.Paragraphs.Last.Alignment = wdAlignParagraphRight
```

▶ [As it applies to the **Words** object.](#)

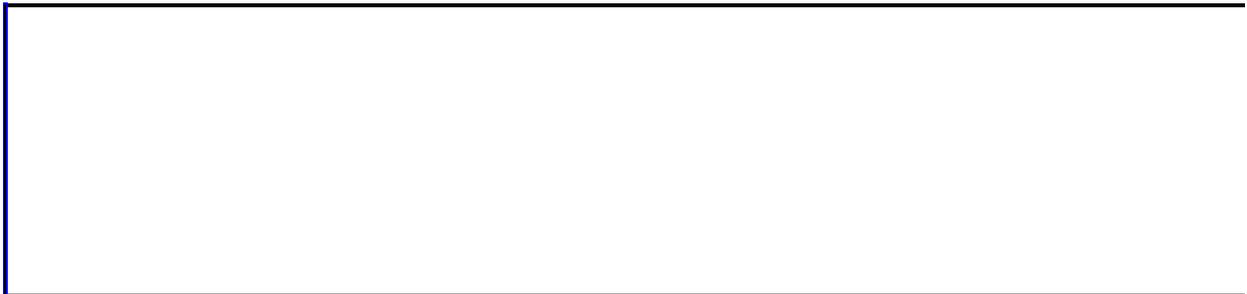
This example applies bold formatting to the last word in the selection.

```
If Selection.Words.Count >= 2 Then  
    Selection.Words.Last.Bold = True  
End If
```

▶ [As it applies to the **Rows** object.](#)

This example deletes the last row in table one.

```
ActiveDocument.Tables(1).Rows.Last.Cells.Delete
```



LastChild Property

Returns a [DiagramNode](#) object that represents the last child node of a parent node.

expression.**LastChild**

expression Required. An expression that returns a [DiagramNodeChildren](#) object.

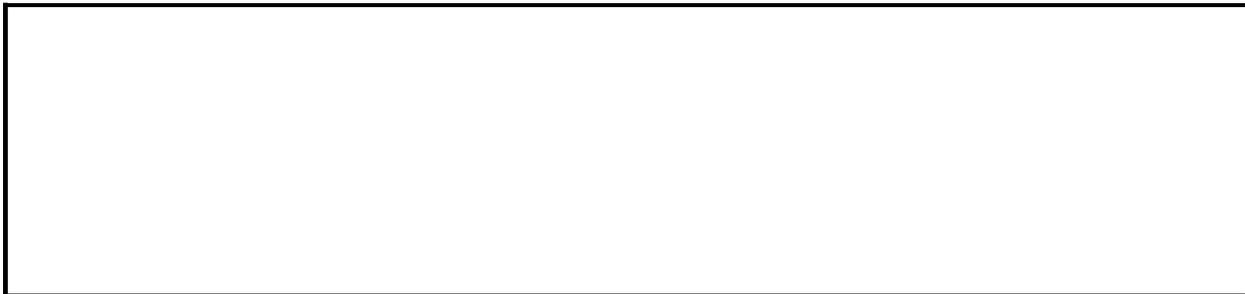
Remarks

Use the [FirstChild](#) property to access the first child node in a diagram. Use the [Root](#) property to access the parent node in a diagram.

Example

This example adds an organization chart diagram to the current document, adds three nodes, and assigns the first and last diagram nodes to variables.

```
Sub FirstChild()  
    Dim shpDiagram As Shape  
    Dim dgnRoot As DiagramNode  
    Dim dgnFirstChild As DiagramNode  
    Dim dgnLastChild As DiagramNode  
    Dim intCount As Integer  
  
    'Add organization chart to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add the first diagram node to the organization chart  
    Set dgnRoot = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three diagram child nodes under the first diagram node  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next  
  
    'Assign the first and last child nodes to variables  
    Set dgnFirstChild = dgnRoot.Children.FirstChild  
    Set dgnLastChild = dgnRoot.Children.LastChild  
End Sub
```



LastRecord Property

Returns or sets the number of the last data record to be merged in a mail merge operation. Read/write **Long**.

expression.**LastRecord**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

Example

This example merges the main document with data records 2 through 4 and sends the merge documents to a new document.

```
With ActiveDocument.MailMerge
    .DataSource.FirstRecord = 2
    .DataSource.LastRecord = 4
    .Destination = wdSendToNewDocument
    .Execute
End With
```



↳ [Show All](#)

Layout Property

Returns or sets an [MsoOrgChartLayoutType](#) constant to indicate the formatting of the child nodes in an organization chart. Read/write.

MsoOrgChartLayoutType can be one of these MsoOrgChartLayoutType constants.

msoOrgChartLayoutAssistant Places child nodes as assistants.

msoOrgChartLayoutBothHanging Places child nodes vertically below the parent node on both the left and the right side.

msoOrgChartLayoutLeftHanging Places child nodes vertically below the parent node on the left side.

msoOrgChartLayoutMixed Return value for a parent node that has children formatted using more than one **MsoOrgChartLayoutType**.

msoOrgChartLayoutRightHanging Places child nodes vertically below the parent node on the right side.

msoOrgChartLayoutStandard Places child nodes horizontally below the parent node.

expression.Layout

expression Required. An expression that returns a [DiagramNode](#) object.

Example

This example creates an organization chart in the active document with three child nodes and places them vertically beneath the parent node along the right side.

```
Sub OrgChartLayoutHangRight()  
    Dim shpOrgChart As Shape  
    Dim dgnRoot As DiagramNode  
    Dim dgnManagerShape As DiagramNode  
    Dim intCount As Integer  
  
    'Add an org chart to the active document and  
    'add the first (parent) node  
    Set shpOrgChart = ActiveDocument.Shapes.AddDiagram( _  
        Type:=msoDiagramOrgChart, Left:=10, _  
        Top:=15, Width:=400, Height:=475)  
    Set dgnRoot = shpOrgChart.DiagramNode.Children.AddNode  
  
    'Add three child nodes to the parent node  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next  
  
    'Format the child nodes to hang vertically along the  
    'right directly under the parent node.  
    dgnRoot.Layout = msoOrgChartLayoutRightHanging  
End Sub
```



↳ [Show All](#)

LayoutMode Property

Returns or sets the layout mode for the current document.
Read/write [WdLayoutMode](#).

WdLayoutMode can be one of these WdLayoutMode constants.

wdLayoutModeDefault No grid is used to lay out text.

wdLayoutModeGenko Text is laid out on a grid; the user specifies the number of lines and the number of characters per line. As the user types, Microsoft Word automatically aligns characters with gridlines.

wdLayoutModeGrid Text is laid out on a grid; the user specifies the number of lines and the number of characters per line. As the user types, Microsoft Word doesn't automatically align characters with gridlines.

wdLayoutModeLineGrid Text is laid out on a grid; the user specifies the number of lines, but not the number of characters per line.

expression.LayoutMode

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example sets the layout mode for the active document so that Microsoft Word automatically aligns typed text to a grid.

```
ActiveDocument.PageSetup.LayoutMode = wdLayoutModeGenko
```



↳ [Show All](#)

Leader Property

Returns or sets the leader for the specified **TabStop** object. Read/write [WdTabLeader](#).

WdTabLeader can be one of these WdTabLeader constants.

wdTabLeaderDashes

wdTabLeaderDots

wdTabLeaderHeavy

wdTabLeaderLines

wdTabLeaderMiddleDot

wdTabLeaderSpaces

expression.**Leader**

expression Required. An expression that returns a [TabStop](#) object.

Example

This example changes the leader for all tab stops that have a leader to dashes for all the paragraphs in the active document.

```
Dim tsLoop As TabStop
```

```
For each tsLoop in ActiveDocument.Paragraphs.TabStops
```

```
    If tsLoop.Leader <> wdTabLeaderSpaces Then
```

```
        tsLoop.Leader = wdTabLeaderDashes
```

```
    End If
```

```
Next tsLoop
```



↳ [Show All](#)

Left Property

▶ [Left property as it applies to the **Shape** and **ShapeRange** objects.](#)

Returns or sets a **Single** that represents the horizontal position, measured in points, of the specified shape or shape range. Can also be any valid [WdShapePosition](#) constant. Read/write.

WdShapePosition can be one of these WdShapePosition constants.

WdShapeBottom

WdShapeCenter

WdShapeInside

WdShapeLeft

WdShapeOutside

WdShapeRight

WdShapeTop

expression.**Left**

expression Required. An expression that returns one of the above objects.

Remarks

The position of a shape is measured from the upper-left corner of the shape's bounding box to the shape's anchor. The [RelativeHorizontalPosition](#) property controls whether the anchor is positioned alongside a character, column, margin, or the edge of the page.

For a [ShapeRange](#) object that contains more than one shape, the **Left** property sets the horizontal position of each shape.

▶ [Left property as it applies to the Application, Task, and Window objects.](#)

Returns or sets a **Long** that represents the horizontal position of the active document (for the [Application](#) object) or the specified task or window, measured in points. 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 document to 1 inch from the left edge of the page.

```
With ActiveDocument.Shapes(1)
    .RelativeHorizontalPosition = _
        wdRelativeHorizontalPositionPage
    .Left = InchesToPoints(1)
End With
```

This example sets the horizontal position of the first and second shapes in the active document to 1 inch from the left edge of the column.

```
With ActiveDocument.Shapes.Range(Array(1, 2))
    .RelativeHorizontalPosition = _
        wdRelativeHorizontalPositionColumn
    .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 = wdWindowStateNormal
    .Left = 100
    .Top = 0
End With
```



LeftIndent Property

Returns or sets a **Single** that represents the left indent value (in points) for the specified paragraphs, table rows, or HTML division. Read/write.

expression.**LeftIndent**

Example

This example sets the left indent of the first paragraph in the active document to 1 inch. The **InchesToPoints** method is used to convert inches to points.

```
ActiveDocument.Paragraphs(1).LeftIndent = InchesToPoints(1)
```

This example sets the left indent for all rows in the first table in the active document.

```
ActiveDocument.Tables(1).Rows.LeftIndent = InchesToPoints(1)
```



LeftMargin Property

Returns or sets the distance (in points) between the left edge of the page and the left boundary of the body text. Read/write **Single**.

expression.**LeftMargin**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

If the [MirrorMargins](#) property is set to **True**, the **LeftMargin** property controls the setting for inside margins and the [RightMargin](#) property controls the setting for outside margins.

Example

This example sets the left margin to 1 inch (72 points) for the second section in the active document.

```
ActiveDocument.Sections(2).PageSetup.LeftMargin = 72
```



LeftPadding Property

Returns or sets the amount of space (in points) to add to the left of the contents of a single cell or all the cells in a table. Read/write **Single**.

expression.**LeftPadding**

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

Remarks

The setting of the **LeftPadding** property for a single cell overrides the setting of the **LeftPadding** property for the entire table.

Example

This example sets the left padding for the first table in the active document to 40 pixels.

```
ActiveDocument.Tables(1).LeftPadding = _  
    PixelsToPoints(40, False)
```



Length Property

-

When the [AutoLength](#) property of the specified callout is set to **False**, the **Length** property returns the length (in points) of the first segment of the callout line (the segment attached to the text callout box). Applies only to callouts whose lines consist of more than one segment (types **msoCalloutThree** and **msoCalloutFour**). Read-only **Single**.

expression.**Length**

expression Required. An expression that returns a [CalloutFormat](#) object.

Remarks

This property is read-only. Use the [CustomLength](#) method to set the value of this property for the [CalloutFormat](#) object.

Example

This example specifies that if the first line segment in the callout named "co1" has a fixed length, then the length of the first line segment in the callout named "co2" will also be fixed at that same length. For the example to work, both callouts must have multiple-segment lines.

```
Dim sngLength As Single
```

```
With ActiveDocument.Shapes
```

```
    With .Item("co1").Callout
```

```
        If Not .AutoLength Then sngLength = .Length
```

```
    End With
```

```
    If sngLength Then _
```

```
        .Item("co2").Callout.CustomLength sngLength
```

```
End With
```



Letterhead Property

-
True if space is reserved for a preprinted letterhead in a letter created by the Letter Wizard. Read/write **Boolean**.

Note The [LetterheadSize](#) property controls the size of the reserved letterhead space.

expression.**Letterhead**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, reserves an inch of space at the top of the page for a preprinted letterhead, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim lcNew As LetterContent

Set lcNew = New LetterContent

With lcNew
    .Letterhead = True
    .LetterheadLocation = wdLetterTop
    .LetterheadSize = InchesToPoints(1)
End With
ActiveDocument.RunLetterWizard _
    LetterContent:=lcNew, WizardMode:=True
```



↳ [Show All](#)

LetterheadLocation Property

Returns or sets the location of the preprinted letterhead in a letter created by the Letter Wizard. Read/write [WdLetterheadLocation](#).

WdLetterheadLocation can be one of these WdLetterheadLocation constants.

wdLetterBottom

wdLetterLeft

wdLetterRight

wdLetterTop

expression.**LetterheadLocation**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, reserves an inch of space at the top of the page for a preprinted letterhead, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim lcNew As LetterContent

Set lcNew = New LetterContent

With lcNew
    .Letterhead = True
    .LetterheadLocation = wdLetterTop
    .LetterheadSize = InchesToPoints(1)
End With

ActiveDocument.RunLetterWizard LetterContent:=lcNew
```



LetterheadSize Property

Returns or sets the amount of space (in points) to be reserved for a preprinted letterhead in a letter created by the Letter Wizard. Read/write **Single**.

expression.**LetterheadSize**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example retrieves the Letter Wizard elements from the active document, changes the preprinted letterhead settings, and then uses the [SetLetterContent](#) method to update the active document to reflect the changes.

```
Set myLetterContent = ActiveDocument.GetLetterContent
With myLetterContent
    .Letterhead = True
    .LetterheadLocation = wdLetterTop
    .LetterheadSize = InchesToPoints(1.5)
End With
ActiveDocument.SetLetterContent LetterContent:=myLetterContent
```



↳ [Show All](#)

LetterStyle Property

Returns or sets the layout of a letter created by the Letter Wizard. Read/write [WdLetterStyle](#).

WdLetterStyle can be one of these WdLetterStyle constants.

wdFullBlock

wdModifiedBlock

wdSemiBlock

expression.LetterStyle

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, selects a letter style, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set aLetterContent = New LetterContent
aLetterContent.LetterStyle = wdFullBlock
ActiveDocument.RunLetterWizard _
    LetterContent:=aLetterContent, WizardMode:=True
```



↳ [Show All](#)

Level Property

▸ [Level property as it applies to the **HeadingStyle** object.](#)

Returns or sets the level for the heading style in a table of contents or table of figures. Read/write **Integer**.

expression.**Level**

expression Required. An expression that returns a [HeadingStyle](#) object.

▸ [Level property as it applies to the **Subdocument** object.](#)

Returns the heading level used to create the subdocument. Read-only **Long**.

expression.**Level**

expression Required. An expression that returns a [Subdocument](#) object.

Example

► [As it applies to the **HeadingStyle** object.](#)

This example adds a table of contents at the insertion point in the active document, and then it changes the levels for the heading styles.

```
ActiveDocument.TablesOfContents.Add _
    Range:=Selection.Range, _
    RightAlignPageNumbers:=True, _
    UseHeadingStyles:=True, _
    UpperHeadingLevel:=1, _
    LowerHeadingLevel:=3, _
    IncludePageNumbers:=True, _
    TableID:=wdTOCFormal
With ActiveDocument.TablesOfContents(1).HeadingStyles
    .Add Style:="Title", Level:=1
    .Add Style:="SubTitle", Level:=2
    .Add Style:="List Bullet", Level:=3
End With
With ActiveDocument.TablesOfContents(1)
    .HeadingStyles(1).Level = 2
    .HeadingStyles(2).Level = 4
    .HeadingStyles(3).Level = 6
End With
```

► [As it applies to the **Subdocument** object.](#)

This example looks through each subdocument in the active document and displays the subdocument's heading level.

```
i = 1
If ActiveDocument.Subdocuments.Count > = 1 Then
    For each s in ActiveDocument.Subdocuments
        MsgBox "The heading level for SubDoc " & i _
            & " is " & s.Level
        i = i + 1
    Next s
Else
    MsgBox "There are no subdocuments defined."
End If
```



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.) Read-only.

Example

This example adds a blue dashed line to myDocument.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

This example adds a cross to myDocument and then sets its border to be 8 points thick and red.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeCross, 10, 10, 50, 70).Line
    .Weight = 8
    .ForeColor.RGB = RGB(255, 0, 0)
End With
```



LineBetween Property

True if vertical lines appear between all the columns in the [TextColumns](#) collection. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

expression.**LineBetween**

expression Required. An expression that returns a **TextColumns** collection object.

Example

This example cycles through each section in the active document and displays a message box if the text columns in the section are separated by vertical lines.

```
i = 1
For each s in ActiveDocument.Sections
    If s.PageSetup.TextColumns.LineBetween = True Then
        MsgBox "The columns in section " & i & " contain lines."
    End If
    i = i + 1
Next s
```



LineNumbering Property

Returns or sets the [LineNumbering](#) object that represents the line numbers for the specified [PageSetup](#) object.

expression.**LineNumbering**

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

Remarks

You must be in print layout view to see line numbering.

Example

This example enables line numbering for the active document.

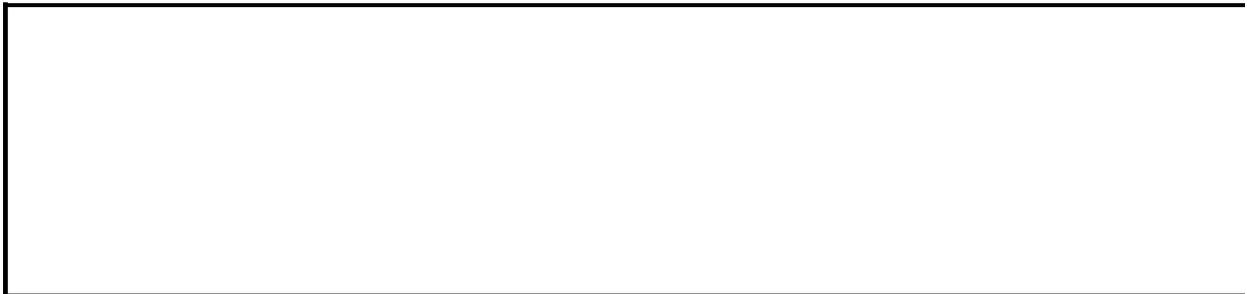
```
ActiveDocument.PageSetup.LineNumbering.Active = True
```

This example enables line numbering for a document named "MyDocument.doc" The starting number is set to one, every fifth line number is shown, and the numbering is continuous throughout all sections in the document.

```
set myDoc = Documents("MyDocument.doc")
With myDoc.PageSetup.LineNumbering
    .Active = True
    .StartingNumber = 1
    .CountBy = 5
    .RestartMode = wdRestartContinuous
End With
```

This example sets the line numbering in the active document equal to the line numbering in MyDocument.doc.

```
ActiveDocument.PageSetup.LineNumbering = Documents("MyDocument.doc")
    .PageSetup.LineNumbering
```



↳ [Show All](#)

LineSpacing Property

-

Returns or sets the line spacing (in points) for the specified paragraphs.
Read/write **Single**.

Remarks

▶ [The **LineSpacing** property can be set after the **LineSpacingRule** property has been set to:](#)

wdLineSpaceAtLeast the line spacing can be greater than or equal to, but never less than, the specified **LineSpacing** value.

wdLineSpaceExactly the line spacing never changes from the specified **LineSpacing** value, even if a larger font is used within the paragraph.

wdLineSpaceMultiple a **LineSpacing** property value must be specified, in points.

Use the **LinesToPoints** method to convert a number of lines to the corresponding value in points. For example, `LinesToPoints(2)` returns the value 24.

Example

This example sets the line spacing for the first paragraph in the active document to always be at least 12 points.

```
With ActiveDocument.Paragraphs(1)
    .LineSpacingRule = wdLineSpaceAtLeast
    .LineSpacing = 12
End With
```

This example triple-spaces the lines in the selected paragraphs.

```
With Selection.Paragraphs
    .LineSpacingRule = wdLineSpaceMultiple
    .LineSpacing = LinesToPoints(3)
End With
```



↳ [Show All](#)

LineSpacingRule Property

Returns or sets the line spacing for the specified paragraphs. Read/write [WdLineSpacing](#).

WdLineSpacing can be one of these WdLineSpacing constants.

wdLineSpace1pt5

wdLineSpaceAtLeast

wdLineSpaceDouble

wdLineSpaceExactly

wdLineSpaceMultiple

wdLineSpaceSingle

expression.**LineSpacingRule**

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

Remarks

Use **wdLineSpaceSingle**, **wdLineSpace1pt5**, or **wdLineSpaceDouble** to set the line spacing to one of these values. To set the line spacing to an exact number of points or to a multiple number of lines, you must also set the **LineSpacing** property.

Example

This example double-spaces the lines in the first paragraph of the active document.

```
ActiveDocument.Paragraphs(1).LineSpacingRule = _  
    wdLineSpaceDouble
```

This example returns the line spacing rule used for the first paragraph in the selection.

```
lrule = Selection.Paragraphs(1).LineSpacingRule
```



LinesPage Property

Returns or sets the number of lines per page in the document grid. Read/write **Single**.

expression.**LinesPage**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example sets the number of lines per page to 35 for the active document.

```
ActiveDocument.PageSetup.LinesPage = 35
```



LinesToDrop Property

Returns or sets the height (in lines) of the specified dropped capital letter.
Read/write **Long**.

expression.**LinesToDrop**

expression Required. An expression that returns a [DropCap](#) object.

Example

This example formats the first character in the active document as a dropped capital letter with a height of three lines.

```
With ActiveDocument.Paragraphs(1).DropCap  
    .Enable  
    .Position = wdDropNormal  
    .LinesToDrop = 3  
End With
```



↳ [Show All](#)

LineStyle Property

Returns or sets the border line style for the specified object. Read/write [WdLineStyle](#).

WdLineStyle can be one of these WdLineStyle constants.

wdLineStyleDashDot

wdLineStyleDashDotDot

wdLineStyleDashDotStroked

wdLineStyleDashLargeGap

wdLineStyleDashSmallGap

wdLineStyleDot

wdLineStyleDouble

wdLineStyleDoubleWavy

wdLineStyleEmboss3D

wdLineStyleEngrave3D

wdLineStyleInset

wdLineStyleNone

wdLineStyleOutset

wdLineStyleSingle

wdLineStyleSingleWavy

wdLineStyleThickThinLargeGap

wdLineStyleThickThinMedGap

wdLineStyleThickThinSmallGap

wdLineStyleThinThickLargeGap

wdLineStyleThinThickMedGap

wdLineStyleThinThickSmallGap

wdLineStyleThinThickThinLargeGap

wdLineStyleThinThickThinMedGap

wdLineStyleThinThickThinSmallGap

wdLineStyleTriple

expression.**LineStyle**

expression Required. An expression that returns a [Border](#) object.

Remarks

Setting the **LineStyle** property for a range that refers to individual characters or words applies a character border.

Setting the **LineStyle** property for a paragraph or range of paragraphs applies a paragraph border. Use the [InsideLineStyle](#) property to apply a border between consecutive paragraphs.

Setting the **LineStyle** property for a section applies a page border around the pages in the section.

Example

If the selection is a paragraph or a collapsed selection, this example adds a single 0.75-point paragraph border above the selection. If the selection doesn't include a paragraph, a border is applied around the selected text.

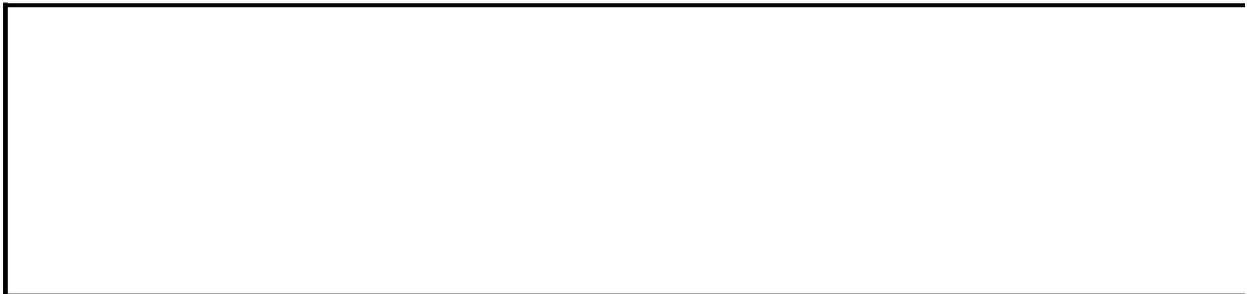
```
With Selection.Borders(wdBorderTop)
    .LineStyle = wdLineStyleSingle
    .LineWidth = wdLineWidth075pt
End With
```

This example adds a double 1.5-point border below each frame in the active document.

```
For Each aFrame In ActiveDocument.Frames
    With aFrame.Borders(wdBorderBottom)
        .LineStyle = wdLineStyleDouble
        .LineWidth = wdLineWidth150pt
    End With
Next aFrame
```

The following example applies a border around the fourth word in the active document. Applying a single border (in this example, a top border) to text applies a border around the text.

```
ActiveDocument.Words(4).Borders(wdBorderTop) _
    .LineStyle = wdLineStyleSingle
```



LineUnitAfter Property

Returns or sets the amount of spacing (in gridlines) after the specified paragraphs. Read/write **Single**.

expression.**LineUnitAfter**

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

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets the spacing after the first paragraph in the active document to one gridline.

```
ActiveDocument.Paragraphs(1).LineUnitAfter = 1
```



LineUnitBefore Property

Returns or sets the amount of spacing (in gridlines) before the specified paragraphs. Read/write **Single**.

expression.**LineUnitBefore**

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

Remarks

For more information on using Microsoft Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets the spacing before the second paragraph in the active document to one gridline.

```
ActiveDocument.Paragraphs(2).LineUnitBefore = 1
```



↳ [Show All](#)

LineWidth Property

Returns or sets the line width of an object's border. Returns a [WdLineWidth](#) constant or **wdUndefined** if the object either has no borders or has borders with more than one line width. Read/write.

WdLineWidth can be one of these WdLineWidth constants.

wdLineWidth025pt

wdLineWidth050pt

wdLineWidth075pt

wdLineWidth100pt

wdLineWidth150pt

wdLineWidth225pt

wdLineWidth300pt

wdLineWidth450pt

wdLineWidth600pt

expression.**LineWidth**

expression Required. An expression that returns a [Border](#) object.

Remarks

If the specified line width isn't available for the border's line style, this property generates an error. To determine the line widths available for a particular line style, see the **Borders and Shading** dialog box (**Format** menu).

Example

This example adds a border below the first row in the first table of the active document.

```
If ActiveDocument.Tables.Count >= 1 Then
    With ActiveDocument.Tables(1).Rows(1).Borders(wdBorderBottom)
        .LineStyle = wdLineStyleSingle
        .LineWidth = wdLineWidth050pt
    End With
End If
```

This example adds a wavy, red line to the left of the selection.

```
With Selection.Borders(wdBorderLeft)
    .LineStyle = wdLineStyleSingleWavy
    .LineWidth = wdLineWidth075pt
    .ColorIndex = wdRed
End With
```



LinkedStyle Property

Returns or sets the name of the style that's linked to the specified [ListLevel](#) object. Read/write **String**.

expression.**LinkedStyle**

expression Required. An expression that returns a [ListLevel](#) object.

Example

This example sets the variable `myListTemp` to the first list template (excluding **None**) on the **Outline Numbered** tab in the **Bullets and Numbering** dialog box (**Format** menu). Each level in the list has a matching heading style linked to it.

```
Set myListTemp = _  
    ListGalleries(wdOutlineNumberGallery).ListTemplates(1)  
For Each mylevel In myListTemp.ListLevels  
    mylevel.LinkedStyle = "Heading " & mylevel.index  
Next mylevel
```



LinkFormat Property

-

Returns a [LinkFormat](#) object that represents the link options of the specified field, inline shape, or shape that's linked to a file. Read/only.

Example

This example inserts a graphic as an inline shape (using an INCLUDEPICTURE field) and then displays the source name (Tiles.bmp).

```
Set iShape = ActiveDocument.InlineShapes _  
    .AddPicture(FileName:="C:\windows\Tiles.bmp", _  
    LinkToFile:=True, SaveWithDocument:=False, _  
    Range:=Selection.Range)  
MsgBox iShape.LinkFormat.SourceName
```

This example updates any fields in the active document that aren't updated automatically.

```
For Each afield In ActiveDocument.Fields  
    If afield.LinkFormat.AutoUpdate = False _  
        Then afield.LinkFormat.Update  
Next afield
```



LinkStyle Property

-
Sets or returns a **Variant** that represents a link between a paragraph and a character style. Read/write.

expression.**LinkStyle**

expression Required. An expression that returns a [Style](#) object.

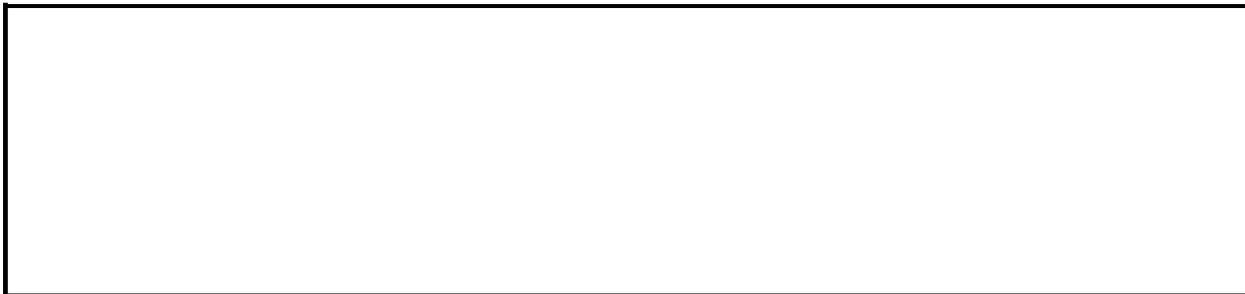
Remarks

When a character style and a paragraph style are linked, the two styles take on the same character formatting.

Example

This example creates and formats a new character style, and then it links the character style to the built-in heading style "Heading 1" so that the "Heading 1" style takes on the character formatting of the newly added style.

```
Sub LinkHeadStyle()  
    Dim styChar1 As Style  
  
    Set styChar1 = ActiveDocument.Styles.Add(Name:="Heading 1 Charac  
        Type:=wdStyleTypeCharacter)  
    With styChar1  
        .Font.Name = "Verdana"  
        .Font.Bold = True  
        .Font.Shadow = True  
        With .Font.Borders(1)  
            .LineStyle = wdLineStyleDot  
            .LineWidth = wdLineWidth300pt  
            .Color = wdColorDarkRed  
        End With  
    End With  
    ActiveDocument.Styles("Heading 1").LinkStyle = ActiveDocument _  
        .Styles("Heading 1 Characters")  
  
    With ActiveDocument.Content  
        .InsertParagraphAfter  
        .InsertAfter "New Linked Style"  
        .Select  
    End With  
  
    Selection.Collapse Direction:=wdCollapseEnd  
    Selection.Style = ActiveDocument.Styles("Heading 1")  
  
End Sub
```



LinkToPrevious Property

-
True if the specified header or footer is linked to the corresponding header or footer in the previous section. When a header or footer is linked, its contents are the same as in the previous header or footer. Read/write **Boolean**.

expression.**LinkToPrevious**

expression Required. An expression that returns a [HeaderFooter](#) object.

Remarks

Because the **LinkToPrevious** property is set to **True** by default, you can add headers, footers, and page numbers to your entire document by working with the headers, footers, and page numbers in the first section. For instance, the following example adds page numbers to the header on all pages in all sections of the active document.

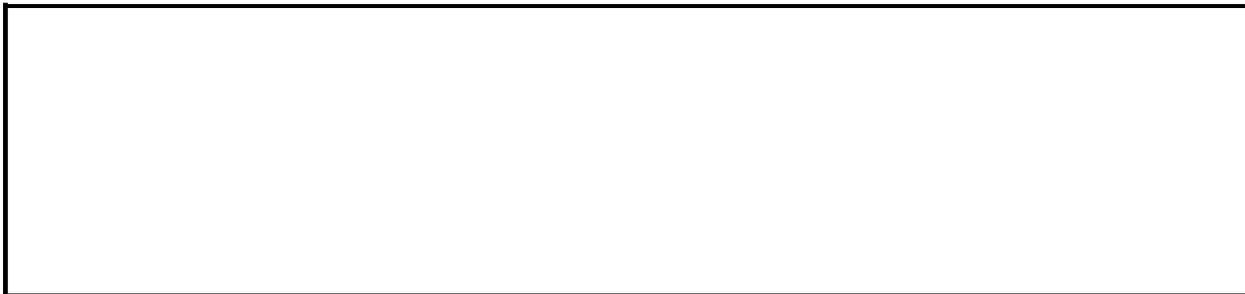
```
ActiveDocument.Sections(1) _  
    .Headers(wdHeaderFooterPrimary).PageNumbers.Add
```

The **LinkToPrevious** property applies to each header or footer individually. For example, the **LinkToPrevious** property could be set to **True** for the even-numbered-page header but **False** for the even-numbered-page footer.

Example

The first part of this example creates a new document with two sections. The second part creates unique headers for even-numbered and odd-numbered pages in sections one and two in the new document.

```
Documents.Add
With Selection
    For j = 1 to 4
        .TypeParagraph
        .InsertBreak
        .TypeParagraph
    Next j
End With
With ActiveDocument
    .Paragraphs(5).Range.InsertBreak Type:=wdSectionBreakNextPage
    .PageSetup.OddAndEvenPagesHeaderFooter = True
End With
With ActiveDocument.Sections(2)
    With .Headers(wdHeaderFooterPrimary)
        .LinkToPrevious = False
        .Range.InsertBefore "Section 2 Odd Header"
    End With
    With .Headers(wdHeaderFooterEvenPages)
        .LinkToPrevious = False
        .Range.InsertBefore "Section 2 Even Header"
    End With
End With
With ActiveDocument.Sections(1)
    .Headers(wdHeaderFooterPrimary) _
        .Range.InsertBefore "Section 1 Odd Header"
    .Headers(wdHeaderFooterEvenPages) _
        .Range.InsertBefore "Section 1 Even Header"
End With
```



List Property

Returns a [List](#) object that represents the first formatted list contained in the specified [ListFormat](#) object.

expression.**List**

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

Remarks

If the first paragraph in the range for the **ListFormat** object is not formatted as a list, the **List** property returns nothing.

Example

This example returns the first list in the selection, and then it applies the first list template (excluding **None**) on the **Numbered** tab in the **Bullets and Numbering** dialog box (**Format** menu). The selection can only contain one list.

```
Set mylist = Selection.Range.ListFormat.List  
mylist.ApplyListTemplate _  
    ListTemplate:=ListGalleries(wdNumberGallery) _  
    .ListTemplates(1)
```



ListEntries Property

Returns a [ListEntries](#) collection that represents all the items in a [DropDown](#) object.

expression.**ListEntries**

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

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example retrieves the text of the active item from the drop-down form field named "DropDown1."

```
Set myField = ActiveDocument.FormFields("DropDown1").DropDown
num = myField.Value
myName = myField.ListEntries(num).Name
```

This example retrieves the total number of items in the active drop-down form field (the document should be protected for forms). If there are two or more items, this example sets the second item as the active item.

```
Set myField = Selection.FormFields(1)
If myfield.Type = wdFieldFormDropDown Then
    num = myField.DropDown.ListEntries.Count
    If num >= 2 Then myField.DropDown.Value = 2
End If
```



ListFormat Property

-

Returns a [ListFormat](#) object that represents all the list formatting characteristics of a range. Read-only.

Example

This example sets the variable `myDoc` to a range that includes paragraphs three through six of the active document. The example then either applies the default outline-numbered list format to the range or removes it, depending on whether or not the format was already applied to the range.

```
Set myDoc = ActiveDocument
Set myRange = _
    myDoc.Range(Start:= myDoc.Paragraphs(3).Range.Start, _
    End:=myDoc.Paragraphs(6).Range.End)
myRange.ListFormat.ApplyOutlineNumberDefault
```

This example applies the second list template on the **Numbered** tab in the **Bullets and Numbering** dialog box to all the paragraphs in the selection.

```
Selection.Range.ListFormat.ApplyListTemplate _
    ListTemplate:=ListGalleries(wdNumberGallery).ListTemplates(2)
```



ListGalleries Property

Returns a [ListGalleries](#) collection that represents the three list template galleries (**Bulleted**, **Numbered**, and **Outline Numbered**). Each gallery corresponds to a tab in the **Bullets and Numbering** dialog box (**Format** menu).

expression.**ListGalleries**

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 sets the variable `mylsttmp` to the second list template on the **Outline Numbered** tab in the **Bullets and Numbering** dialog box. The example then applies that template to the first list in the active document.

```
Set mylsttmp = _  
    ListGalleries(wdOutlineNumberGallery).ListTemplates(2)  
ActiveDocument.Lists(1).ApplyListTemplate ListTemplate:=mylsttmp
```

This example cycles through the **ListGalleries** collection and changes the templates in each list template gallery back to the built-in template.

```
For Each listgal In ListGalleries  
    For i = 1 To 7  
        listgal.Reset(i)  
    Next i  
Next listgal
```



↳ [Show All](#)

ListLevelNumber Property

▶ [ListLevelNumber property as it applies to the ListFormat object.](#)

Returns or sets the list level for the first paragraph in the specified [ListFormat](#) object. Read/write **Long**.

expression.**ListLevelNumber**

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

▶ [ListLevelNumber property as it applies to the Style object.](#)

Returns the list level for the specified style. Read-only **Long**.

expression.**ListLevelNumber**

expression Required. An expression that returns a [Style](#) object.

Example

▶ [As it applies to the **ListFormat** object.](#)

This example returns the list level for the third paragraph in the active document.

```
lev = ActiveDocument.Paragraphs(3).Range.ListFormat.ListLevelNumber
```

▶ [As it applies to the **Style** object.](#)

This example displays the list level for the Heading 3 style.

```
Msgbox ActiveDocument.Styles(wdStyleHeading3).ListLevelNumber
```



ListLevels Property

Returns a [ListLevels](#) collection that represents all the levels for the specified [ListTemplate](#).

expression.**ListLevels**

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

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the variable `myListTemp` to the first list template (excluding **None**) on the **Outline Numbered** tab in the **Bullets and Numbering** dialog box (**Format** menu). Each level in the list has a matching heading style linked to it.

```
Set myListTemp = _  
    ListGalleries(wdOutlineNumberGallery).ListTemplates(1)  
For Each mylevel In myListTemp.ListLevels  
    mylevel.LinkedStyle = "Heading " & mylevel.index  
Next mylevel
```



ListParagraphs Property

Returns a [ListParagraphs](#) collection that represents all the numbered paragraphs in the list, document, or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a yellow background to each numbered or bulleted paragraph in the first document.

```
For Each numpar In Documents(1).ListParagraphs
    numpar.Shading.BackgroundPatternColorIndex = wdYellow
Next numpar
```

This example double underlines the paragraphs in the second list in the active document.

```
For Each mypara In ActiveDocument.Lists(2).ListParagraphs
    mypara.Range.Underline = wdUnderlineDouble
Next mypara
```



ListPictureBullet Property

Returns the **InlineShape** object that represents the picture used as a bullet in a picture bullet list.

expression.**ListPictureBullet**

expression Required. An expression that returns a [ListFormat](#) object.

Example

This example sets the height and width of the selected picture bullet. This example assumes that the insertion point in the document is located in a paragraph formatted with a picture bullet.

```
Sub ListPictBullet()  
  With Selection.Range.ListFormat.ListPictureBullet  
    .Width = InchesToPoints(Inches:=0.5)  
    .Height = InchesToPoints(Inches:=0.05)  
  End With  
End Sub
```



Lists Property

-
Returns a [Lists](#) collection that contains all the formatted lists in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example formats the selection as a numbered list. The example then displays a message box that reports the number of lists in the active document.

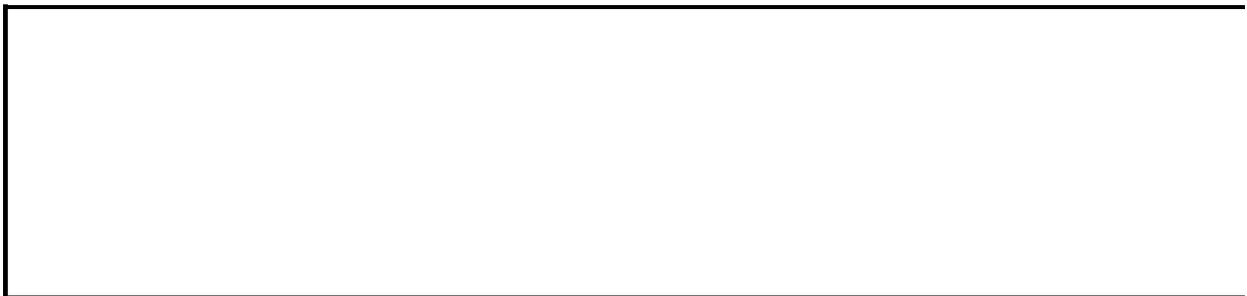
```
Selection.Range.ListFormat.ApplyListTemplate _  
    ListTemplate:=ListGalleries(wdNumberGallery).ListTemplates(2)  
MsgBox "This document has " & ActiveDocument.Lists.Count _  
    & " lists."
```

This example formats the third list in the active document with the default bulleted list format. If the list is already formatted with a bulleted list format, the example removes the formatting.

```
If ActiveDocument.Lists.Count >= 3 Then  
    ActiveDocument.Lists(3).Range.ListFormat.ApplyBulletDefault  
End If
```

This example displays a message box that reports the number of items in each list in MyLetter.doc.

```
Set myDoc = Documents("MyLetter.doc")  
i = myDoc.Lists.Count  
For each li in myDoc.Lists  
    MsgBox "List " & i & " has " & li.CountNumberedItems _  
        & " items."  
    i = i - 1  
Next li
```



ListString Property

Returns a **String** that represents the appearance of the list value of the first paragraph in the range for the specified [ListFormat](#) object. For example, the second paragraph in an alphabetic list would return B. Read-only.

expression.**ListString**

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

Remarks

For a bulleted list, you will need to apply the correct font in order to see the string. Most bullets use the Symbol or Wingdings font.

Use the [ListValue](#) property to return the numeric value of the paragraph.

Example

This example displays both the numeric value of the first paragraph in the selection and the string representation of the list value.

```
v = Selection.Range.ListFormat.ListValue
lstring = Selection.Range.ListFormat.ListString
MsgBox "List value " & v _
    & " is represented by the string " & lstring
```



ListTemplate Property

Returns a [ListTemplate](#) object that represents the list formatting for the specified [Style](#) or [ListFormat](#) object.

expression.**ListTemplate**

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

Remarks

A list template includes all the formatting that defines a particular list. Each of the seven formats (excluding **None**) found on each of the tabs in the **Bullets and Numbering** dialog box (**Format** menu) corresponds to a list template. Documents and templates can also contain collections of list templates.

If the first paragraph in the range for the **ListFormat** object is not formatted as a list, the **ListTemplate** property returns **Nothing**.

Example

This example checks to see which list template is used for the second paragraph in the active document, and then it applies that list template to the selection.

```
Set myltemp = ActiveDocument.Paragraphs(2).Range. _  
    ListFormat.ListTemplate  
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=myltemp
```



ListTemplates Property

Returns a [ListTemplates](#) collection that represents all the list formats for the specified document, template, or list gallery. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the variable `mytemp` to the first list template on the **Outline Numbered** tab in the **Bullets and Numbering** dialog box. The template is modified to use lowercase letters for each level, and it's applied to the second list in the active document.

```
Set mytemp = ListGalleries(wdOutlineNumberGallery).ListTemplates(1)
For each lev in mytemp.ListLevels
    lev.NumberStyle = wdListNumberStyleLowercaseLetter
Next lev
ActiveDocument.Lists(2).ApplyListTemplate ListTemplate:=mytemp
```

This example displays the number of list templates used in the active document.

```
Msgbox ActiveDocument.ListTemplates.Count
```



▾ [Show All](#)

ListType Property

Returns the type of lists that are contained in the range for the specified [ListFormat](#) object. Read only [WdListType](#).

WdListType can be one of these WdListType constants.

wdListBullet

wdListListNumOnly

wdListMixedNumbering

wdListNoNumbering

wdListOutlineNumbering

wdListPictureBullet

wdListSimpleNumbering

expression.ListType

expression Required. An expression that returns a ListFormat.

Remarks

The constant **wdListListNumOnly** refers to LISTNUM fields, which are fields that can be added within the text of a paragraph.

Example

This example checks to see if the first list in the active document is a simple numbered list. If it is, the fourth list template on the **Numbered** tab of the **Bullets and Numbering** dialog box (**Format** menu) is applied.

```
Set myList = ActiveDocument.Lists(1)
If myList.Range.ListFormat.ListType = wdListSimpleNumbering Then
    myList.ApplyListTemplate _
        ListTemplate:=ListGalleries(wdNumberGallery) _
        .ListTemplates(4)
End If
```



ListValue Property

-

Returns the numeric value of the first paragraph in the range for the specified **ListFormat** object. For example, the **ListValue** property applied to the second paragraph in an alphabetic list would return 2. Read-only **Long**.

expression.**ListValue**

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

Remarks

Use the [ListString](#) property to return a string that represents the list value.

If the **ListFormat** object applies to a bulleted list, the **ListValue** property returns 1.

If the **ListFormat** object applies to an outline-numbered list, the **ListValue** property returns the numeric value of the first paragraph as it occurs in the sequence of paragraphs at the same level. For example, if the first paragraph for a specified **ListFormat** object were numbered "A.2," the **ListValue** would return 2.

This property will not return the value for a LISTNUM field.

Example

This example displays both the numeric value of the first paragraph in the selection and the string representation of that value.

```
v = Selection.Range.ListFormat.ListValue  
lstring = Selection.Range.ListFormat.ListString  
MsgBox "List value " & v _  
    & " is represented by the string " & lstring
```



LocalNetworkFile Property

-
True if Microsoft Word creates a local copy of a file on the user's machine when editing a file stored on a network server. Read/write **Boolean**.

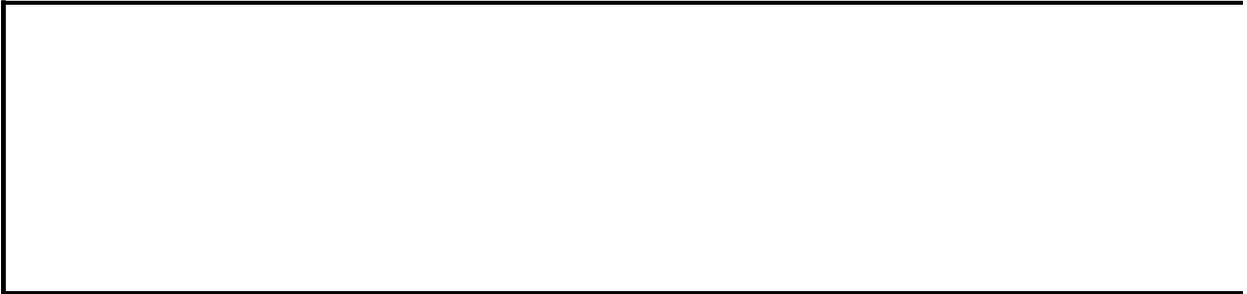
expression.**LocalNetworkFile**

expression Required. An expression that returns an [Options](#) object.

Example

This example instructs Word to not make a local copy of files stored on a server.

```
Sub LocalFile()  
    Application.Options.LocalNetworkFile = False  
End Sub
```



↳ [Show All](#)

Location Property

- ▶ [Location property as it applies to the **EndnoteOptions** and **Endnotes** objects.](#)

Returns or sets the position of all endnotes. Read/write [WdEndnoteLocation](#).

WdEndnoteLocation can be one of these WdEndnoteLocation constants.

wdEndOfDocument

wdEndOfSection

expression.**Location**

expression Required. An expression that returns an [Endnotes](#) or [EndnoteOptions](#) object.

- ▶ [Location property as it applies to the **FootnoteOptions** and **Footnotes** objects.](#)

Returns or sets the position of all footnotes. Read/write [WdFootnoteLocation](#).

WdFootnoteLocation can be one of these WdFootnoteLocation constants.

wdBeneathText

wdBottomOfPage

expression.**Location**

expression Required. An expression that returns a [Footnotes](#) or [FootnoteOptions](#) object.

Example

▶ [As it applies to the **EndnoteOptions** and **Endnotes** objects.](#)

This example positions all endnotes at the end of sections.

```
ActiveDocument.Endnotes.Location = wdEndOfSection
```

[As it applies to the **FootnoteOptions** and **Footnotes** objects.](#)

This example positions footnotes at the bottom of each page.

```
ActiveDocument.Footnotes.Location = wdBottomOfPage
```



↳ [Show All](#)

LockAnchor Property

▶ [LockAnchor property as it applies to the **Frame** object.](#)

True if the specified frame is locked. The frame anchor indicates where the frame will appear in Normal view. You cannot reposition a locked frame anchor.
Read/write **Boolean**.

expression.**LockAnchor**

expression Required. An expression that returns a [Frame](#) object.

▶ [LockAnchor property as it applies to the **Shape** and **ShapeRange** objects.](#)

True if the specified [Shape](#) or [ShapeRange](#) object's anchor is locked to the anchoring range. When a shape has a locked anchor, you cannot move the shape's anchor by dragging it (the anchor doesn't move as the shape is moved).
Read/write **Long**.

expression.**LockAnchor**

expression Required. An expression that returns one of the above objects.

Remarks

A **Shape** object is anchored to a range of text, but you can position it anywhere on the page. The shape is anchored to the beginning of the first paragraph that contains the anchoring range. A shape will always remain on the same page as its anchor.

Example

▶ [As it applies to the **Frame** object.](#)

This example locks the anchor of the first frame in section two of the active document.

```
Set myRange = ActiveDocument.Sections(2).Range
If TypeName(myRange) <> "Nothing" And myRange.Frames.Count > 0 Then
    myRange.Frames(1).LockAnchor = True
End If
```

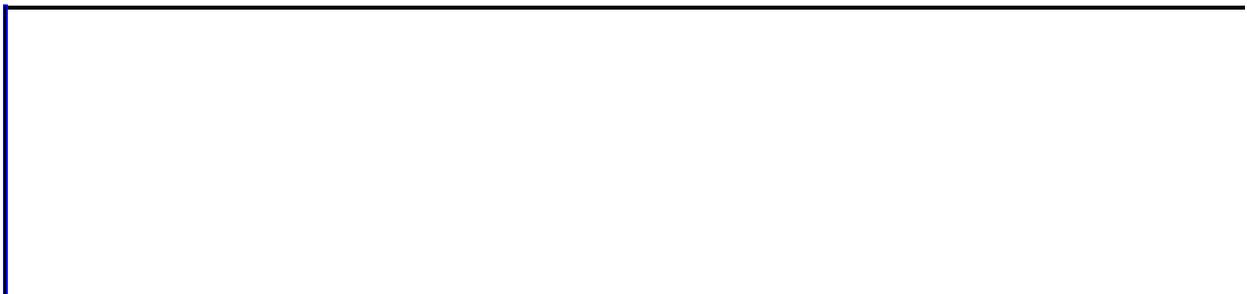
▶ [As it applies to the **Shape** and **ShapeRange** objects.](#)

This example creates a new document, adds a shape to it, and then locks the shape's anchor.

```
Set myDoc = Documents.Add
Set myShape = myDoc.Shapes.AddShape(msoShapeBalloon, _
    100, 100, 140, 70)
myShape.LockAnchor = True
ActiveDocument.ActiveWindow.View.ShowObjectAnchors = True
```

This example returns a message that states the lock status for each shape in the active document.

```
For x = 1 to ActiveDocument.Shapes.Count
    MsgBox "Shape " & x & " is locked - " _
        & ActiveDocument.Shapes(x).LockAnchor
Next x
```



↳ [Show All](#)

LockAspectRatio Property

-
MsoTrue if the specified shape retains its original proportions when you resize it. **MsoFalse** if you can change the height and width of the shape independently of one another when you resize it. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**LockAspectRatio**

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

Example

This example adds a cube to myDocument. The cube can be moved and resized but not reproportioned.

```
Set myDocument = ActiveDocument  
myDocument.Shapes.AddShape(msoShapeCube, _  
    50, 50, 100, 200).LockAspectRatio = msoTrue
```



↳ [Show All](#)

Locked Property

▶ [Locked property as it applies to the **Field**, **LinkFormat**, **MailMergeField**, and **Subdocument** objects.](#)

Subdocument object: **True** if a subdocument in a master document is locked.

LinkFormat object: **True** if a **Field**, **InlineShape**, or **Shape** object is locked to prevent automatic updating. If you use this property with a **Shape** object that's a floating linked picture (a picture added with the **AddPicture** method of the **Shapes** object), an error occurs.

Field or **MailMergeField** object: **True** if the specified field is locked. When a field is locked, you cannot update the field results.

Read/write **Boolean**.

expression.**Locked**

expression Required. An expression that returns one of the above objects.

▶ [Locked property as it applies to the **Fields** object.](#)

True if all fields in the **Fields** collection are locked. Can be **True**, **False**, or **wdUndefined** (if some of the fields in the collection are locked). Read/write **Long**.

expression.**Locked**

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

Example

▶ [As it applies to the **Subdocument** object.](#)

This example checks the first subdocument in the specified master document and sets the master document to allow only comments if the subdocument is locked.

```
If ActiveDocument.Subdocuments(1).Locked = True Then
    ActiveDocument.Protect Type:=wdAllowOnlyComments
End If
```

▶ [As it applies to the **Fields** object.](#)

This example inserts a DATE field at the beginning of the selection and then locks the field.

```
Selection.Collapse Direction:=wdCollapseStart
Set myField = ActiveDocument.Fields.Add(Range:=Selection.Range, _
    Type:=wdFieldDate)
myField.Locked = True
```

This example locks all the fields in the selection.

```
Selection.Fields.Locked = True
```

This example displays a message if some of the fields in the active document are locked.

```
Set theFields = ActiveDocument.Fields
If theFields.Locked = wdUndefined Then
    MsgBox "Some fields are locked"
ElseIf theFields.Locked = False Then
    MsgBox "No fields are locked"
ElseIf theFields.Locked = True Then
    MsgBox "All fields are locked"
End If
```



LowerHeadingLevel Property

-

Returns or sets the ending heading level for a table of contents or table of figures. Corresponds to the ending value used with the \o switch for a Table of Contents (TOC) field. Read/write **Long**.

expression.**LowerHeadingLevel**

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

Remarks

Use the [UpperHeadingLevel](#) property to set the starting heading level. For example, to set the TOC field syntax {TOC \o "1-3"}, set the **LowerHeadingLevel** property to 3 and the **UpperHeadingLevel** property to 1.

Example

This example formats the first table of contents in the active document to show entries formatted with the Heading 2, Heading 3, and Heading 4 styles.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .UseHeadingStyles = True
        .UpperHeadingLevel = 2
        .LowerHeadingLevel = 4
    End With
End If
```



MacroContainer Property

Returns a [Template](#) or [Document](#) object that represents the template or document in which the module that contains the running procedure is stored.

expression.**MacroContainer**

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

Example

This example displays the name of the document or template in which the running procedure is stored.

```
Set cntnr = MacroContainer  
MsgBox cntnr.Name
```



Magenta Property

-

Sets or returns a **Long** that represents the magenta component of a CMYK color.
Read-only.

expression.**Magenta**

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

Example

This example creates a new shape, then retrieves the four CMYK values from an existing shape in the active document, and then sets the CMYK fill color of the new 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.Shapes(1)  
    Set shpStar = ActiveDocument.Shapes.AddShape _  
        (Type:=msoShape5pointStar, Left:=200, _  
         Top:=100, Width:=150, Height:=150)  
  
    'Get current shapes CMYK colors  
    With shpHeart.Fill.ForeColor  
        lngCyan = .Cyan  
        lngMagenta = .Magenta  
        lngYellow = .Yellow  
        lngBlack = .Black  
    End With  
  
    'Set new shape to current shapes CMYK colors  
    shpStar.Fill.ForeColor.SetCMYK _  
        Cyan:=lngCyan, Magenta:=lngMagenta, _  
        Yellow:=lngYellow, Black:=lngBlack  
End Sub
```



Magnifier Property

-
True if the pointer is displayed as a magnifying glass in print preview, indicating that the user can click to zoom in on a particular area of the page or zoom out to see an entire page or spread of pages. Read/write **Boolean**.

expression.**Magnifier**

expression Required. An expression that returns a [View](#) object.

Remarks

This property generates an error if the view is not print preview.

Example

This example switches to print preview and changes the pointer to an insertion point.

```
PrintPreview = True  
ActiveDocument.ActiveWindow.View.Magnifier = False
```



MailAddressFieldName Property

Returns or sets the name of the field that contains e-mail addresses that are used when the mail merge destination is electronic mail. Read/write **String**.

expression.**MailAddressFieldName**

expression Required. An expression that returns a [MailMerge](#) object.

Example

This example merges the document named "FormLetter.doc" with its attached data document and sends the results to the e-mail addresses stored in the Email merge field.

```
With Documents("FormLetter.doc").MailMerge
    .MailAddressFieldName = "Email"
    .MailSubject = "Amazing offer"
    .Destination = wdSendToEmail
    .Execute
End With
```



MailAsAttachment Property

True if the merge documents are sent as attachments when the mail merge destination is an e-mail message or a fax. Read/write **Boolean**.

expression.**MailAsAttachment**

expression Required. An expression that returns a [MailMerge](#) object.

Example

This example performs a mail merge operation and sends the merge results as attachments to e-mail messages. The e-mail addresses are stored in the MailAddress merge field.

```
With Documents("Main.doc").MailMerge
    .MailAsAttachment = True
    .Destination = wdSendToEmail
    .MailSubject = "Special offer"
    .MailAddressFieldName = "MailAddress"
    .Execute
End With
```



MailEnvelope Property

Returns an [MsoEnvelope](#) object that represents an e-mail header for a document.

expression.**MailEnvelope**

expression Required. An expression that returns a [Document](#) object.

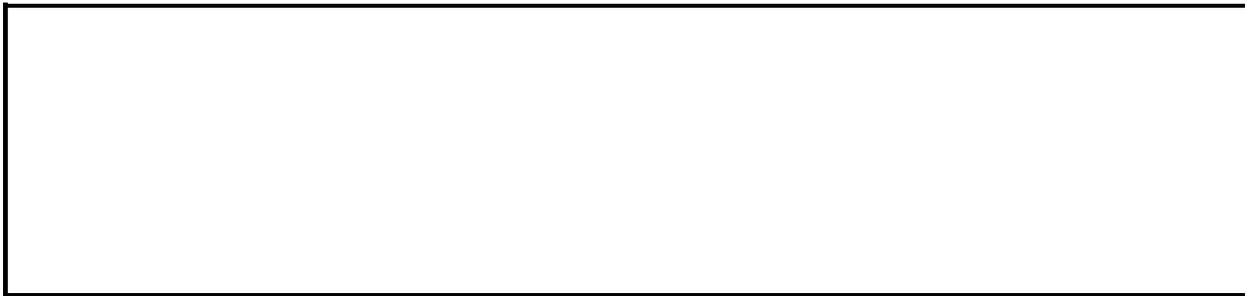
Example

This example sets the comments for the e-mail header of the active document.

```
Sub HeaderComments()
```

```
    ActiveDocument.MailEnvelope.Introduction = _  
        "Please review this document and let me know " & _  
        "what you think. I need your input by Friday." & _  
        " Thanks."
```

```
End Sub
```



↳ [Show All](#)

MailFormat Property

Returns a [WdMailMergeMailFormat](#) constant that represents the format to use when the mail merge destination is an e-mail message. Read/write.

WdMailMergeMailFormat can be one of these WdMailMergeMailFormat constants.

wdMailFormatHTML Sends mail merge e-mail documents using HTML format.

wdMailFormatPlainText Sends mail merge e-mail documents using plain text.

expression.**MailFormat**

expression Required. An expression that returns a [MailMerge](#) object.

Remarks

The **MailFormat** property is ignored if the [MailAsAttachment](#) property is set to **True**.

Example

This example merges the active document to an e-mail message and formats it using HTML.

```
Sub MergeDestination()  
    With ActiveDocument.MailMerge  
        .Destination = wdSendToEmail  
        .MailAsAttachment = False  
        .MailFormat = wdMailFormatHTML  
        .Execute  
    End With  
End Sub
```



MailingInstructions Property

Returns or sets the mailing instruction text for a letter created by the Letter Wizard (for example, "Certified Mail"). Read/write **String**.

expression.**MailingInstructions**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example retrieves the Letter Wizard elements from the active document, changes the text of the mailing instructions, and then uses the [SetLetterContent](#) method to update the active document to reflect the changes.

```
Set myLetterContent = ActiveDocument.GetLetterContent
myLetterContent.MailingInstructions = "Air Mail"
ActiveDocument.SetLetterContent LetterContent:=myLetterContent
```

This example creates a new [LetterContent](#) object, sets several properties (including the mailing instruction text), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
With myContent
    .RecipientReference = "In reply to:"
    .Salutation = "Hello"
    .MailingInstructions = "Certified Mail"
End With
Documents.Add.RunLetterWizard LetterContent:=myContent
```



MailingLabel Property

Returns a [MailingLabel](#) object that represents a mailing label.

expression.**MailingLabel**

expression Required. An expression that returns an [Application](#) object.

Example

This example creates a new Avery 2160 mini-label document for a specified address.

```
addr = "Dave Edson" & vbCr & "123 Skye St." _  
      & vbCr & "Our Town, WA 98004"  
Application.MailingLabel.CreateNewDocument _  
      Name:="2160 mini", Address:=addr, ExtractAddress:=False
```



MailMerge Property

Returns a [MailMerge](#) object that represents the mail merge functionality for the specified document. Read-only.

Note The **MailMerge** object is available regardless of whether the specified document is a mail merge main document. Use the **State** property to determine the current state of the mail merge operation.

Example

This example executes a mail merge if the active document is a main document with an attached data source.

```
Set myMerge = ActiveDocument.MailMerge  
If myMerge.State = wdMainAndDataSource Then myMerge.Execute
```

This example merges the main document with data records 1 through 4 and sends the merge documents to the printer.

```
With ActiveDocument.MailMerge  
    .DataSource.FirstRecord = 1  
    .DataSource.LastRecord = 4  
    .Destination = wdSendToPrinter  
    .SuppressBlankLines = True  
    .Execute  
End With
```



MailMergeDataView Property

True if mail merge data is displayed instead of mail merge fields in the specified window. Read/write **Boolean**.

expression.**MailMergeDataView**

expression Required. An expression that returns a [View](#) object.

Remarks

If the specified window isn't a main document, an error occurs.

Example

If the active document includes at least one mail merge field, this example displays mail merge data from the first record in the attached data source.

```
If ActiveDocument.MailMerge.Fields.Count >= 1 Then
    ActiveDocument.MailMerge.DataSource.ActiveRecord = 1
    ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
    ActiveDocument.ActiveWindow.View.MailMergeDataView = True
End If
```

This example toggles between viewing mail merge fields and viewing the resulting data.

```
With ActiveDocument.ActiveWindow.View
    .ShowFieldCodes = False
    .MailMergeDataView = Not .MailMergeDataView
End With
```



MailMessage Property

Returns a [MailMessage](#) object that represents the active e-mail message.

expression.**MailMessage**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays the **Select Names** dialog box for the active e-mail message.

```
Application.MailMessage.DisplaySelectNamesDialog
```



MailSubject Property

Returns or sets the subject line used when the mail merge destination is electronic mail. Read/write **String**.

expression.**MailSubject**

expression Required. An expression that returns a [MailMerge](#) object.

Example

This example merges the document named "Offer.doc" with its attached data document. The results are sent to the e-mail addresses stored in the EmailNames merge field, and the subject of the mail message is "Amazing Offer."

```
With Documents("Offer.doc").MailMerge
    .MailAddressFieldName = "EmailNames"
    .MailSubject = "Amazing Offer"
    .Destination = wdSendToEmail
    .Execute
End With
```



MailSystem Property

Returns the mail system (or systems) installed on the host machine. Read-only [WdMailSystem](#).

WdMailSystem can be one of these WdMailSystem constants.

wdMAPI

wdNoMailSystem

wdMAPIandPowerTalk

wdPowerTalk

expression.**MailSystem**

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

Remarks

Some of the constants listed above are available only in Microsoft Office Macintosh Edition. For additional information about these constants, consult the language reference Help included with Microsoft Office Macintosh Edition.

Example

This example displays the mail system installed on the host machine.

```
ms = Application.MailSystem  
If ms <> wdNoMailSystem Then  
    MsgBox "This machine has a mail system installed."  
Else  
    MsgBox "This machine has no mail system installed."  
End If
```



↳ [Show All](#)

MainDocumentType Property

Returns or sets the mail merge main document type. Read/write [WdMailMergeMainDocType](#).

WdMailMergeMainDocType can be one of these WdMailMergeMainDocType constants.

wdCatalog

wdDirectory

wdEMail

wdEnvelopes

wdFax

wdFormLetters

wdMailingLabels

wdNotAMergeDocument

expression.MainDocumentType

expression Required. An expression that returns a [MailMerge](#) object.

Remarks

If you set this property for a document that's already a main document, the attached data source is removed.

Example

This example creates a new document and makes it a catalog main document for a mail merge operation.

```
Set myDoc = Documents.Add  
myDoc.MailMerge.MainDocumentType = wdCatalog
```

This example determines whether the active document is a main document for a mail merge operation, and then it displays a message in the status bar.

```
Set doc = ActiveDocument  
If doc.MailMerge.MainDocumentType = wdNotAMergeDocument Then  
    StatusBar = "Not a mail merge main document"  
Else  
    StatusBar = "Document is a mail merge main document."  
End If
```



MAPIAvailable Property

-
True if MAPI is installed. Read-only **Boolean**.

expression.**MAPIAvailable**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays a message if MAPI is installed.

```
If Application.MAPIAvailable = True Then  
    MsgBox "MAPI is available"  
End If
```



MapPaperSize Property

-

True if documents formatted for another country's/region's standard paper size (for example, A4) are automatically adjusted so that they're printed correctly on your country's/region's standard paper size (for example, Letter). Read/write **Boolean**.

expression.**MapPaperSize**

expression Required. An expression that returns an [Options](#) object.

Remarks

This property affects only the printout of your document; its formatting is left unchanged.

Example

This example allows Microsoft Word to adjust paper size according to the country/region setting.

```
Options.MapPaperSize = True
```

This example returns the status of the **Allow A4/Letter paper resizing** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.MapPaperSize
```



↳ [Show All](#)

MappedDataFields Property

Returns a **MappedDataFields** object that represents the [mapped data fields](#) available in Microsoft Word.

expression.**MappedDataFields**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

Example

This example creates a tabbed list of the mapped data fields available in Word and the fields in the data source to which they are mapped. This example assumes that the current document is a mail merge document.

```
Sub MappedFields()  
    Dim intCount As Integer  
    Dim docCurrent As Document  
    Dim docNew As Document  
  
    On Error Resume Next  
  
    Set docCurrent = ThisDocument  
    Set docNew = Documents.Add  
  
    'Add leader tab to new document  
    docNew.Paragraphs.TabStops.Add _  
        Position:=InchesToPoints(3.5), _  
        Leader:=wdTabLeaderDots  
  
    With docCurrent.MailMerge.DataSource  
  
        'Insert heading paragraph for tabbed columns  
        docNew.Content.InsertAfter "Word Mapped Data Field" _  
            & vbTab & "Data Source Field"  
  
        Do  
            intCount = intCount + 1  
  
            'Insert Word mapped data field name and the  
            'corresponding data source field name  
            docNew.Content.InsertAfter .MappedDataFields( _  
                Index:=intCount).Name & vbTab & _  
                .MappedDataFields(Index:=intCount) _  
                .DataFieldName  
  
            'Insert paragraph  
            docNew.Content.InsertParagraphAfter  
        Loop Until intCount = .MappedDataFields.Count  
  
    End With  
  
End Sub
```



MarginBottom Property

Returns or sets the distance (in points) between the bottom of the text frame and the bottom of the inscribed rectangle of the shape that contains the text.

Read/write **Single**.

expression.**MarginBottom**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example adds a rectangle to myDocument, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    0, 0, 250, 140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```



MarginLeft Property

Returns or sets the distance (in points) between the left edge of the text frame and the left edge of the inscribed rectangle of the shape that contains the text. Read/write **Single**.

expression.**MarginLeft**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example adds a rectangle to myDocument, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    0, 0, 250, 140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```



MarginRight Property

-

Returns or sets the distance (in points) between the right edge of the text frame and the right edge of the inscribed rectangle of the shape that contains the text. Read/write **Single**.

expression.**MarginRight**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example adds a rectangle to myDocument, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    0, 0, 250, 140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```



MarginTop Property

-

Returns or sets the distance (in points) between the top of the text frame and the top of the inscribed rectangle of the shape that contains the text. Read/write **Single**.

expression.**MarginTop**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example adds a rectangle to myDocument, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    0, 0, 250, 140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```



MarkComments Property

True if Microsoft Word marks the user's comments in e-mail messages.
Read/write **Boolean**.

expression.**MarkComments**

expression Required. An expression that returns an [EMailOptions](#) object.

Remarks

This property marks comments with the value of the [MarkCommentsWith](#) property. The default value of the **MarkCommentsWith** property is the value of the [UserName](#) property.

Example

This example sets Word to mark comments in e-mail messages with the initials "WK."

```
Application.EmailOptions.MarkCommentsWith = "WK"  
Application.EmailOptions.MarkComments = True
```



MarkCommentsWith Property

Returns or sets the string with which Microsoft Word marks comments in e-mail messages. Read/write **String**.

expression.**MarkCommentsWith**

expression Required. An expression that returns an [EMailOptions](#) object.

Remarks

The default value is the value of the [UserName](#) property.

Example

This example sets Word to mark comments in e-mail messages with the initials "WK."

```
Application.EmailOptions.MarkCommentsWith = "WK"  
Application.EmailOptions.MarkComments = True
```



MatchAlefHamza Property

-
True if find operations match text with matching alef hamzas in an Arabic language document. Read/write **Boolean**.

expression.**MatchAlefHamza**

expression Required. An expression that returns a [Find](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the current find operation to match alef hamzas.

```
Selection.Find.MatchAlefHamza = True
```



MatchAllWordForms Property

-
True if all forms of the text to find are found by the find operation (for instance, if the text to find is "sit," "sat" and "sitting" are found as well). Read/write **Boolean**.

expression.**MatchAllWordForms**

expression Required. An expression that returns a [Find](#) object.

Remarks

Use the [Text](#) property of the [Find](#) object or the *FindText* argument with the [Execute](#) method to specify the text to be searched for in a document.

Example

This example selects the next form of the word "sit" found in the selection or displays a message box if a form of "sit" isn't found.

```
With Selection.Find
    .MatchAllWordForms = True
    .Text = "sit"
    .Execute Format:=False
    If .Found = False Then MsgBox "Not Found"
End With
```



MatchByte Property

-

True if Microsoft Word distinguishes between full-width and half-width letters or characters during a search. Read/write **Boolean**.

Example

This example searches for the term "マイクロソフト" in the specified range without distinguishing between full-width and half-width characters.

```
With Selection.Find
    .ClearFormatting
    .MatchWholeWord = True
    .MatchByte = False
    .Execute FindText:="マイクロソフト"
End With
```



MatchCase Property

-
True if the find operation is case sensitive. The default is **False**. Read/write **Boolean**.

expression.**MatchCase**

expression Required. An expression that returns a [Find](#) object.

Remarks

Use the [Text](#) property of the [Find](#) object or use the *FindText* argument with the [Execute](#) method to specify the text to be located in a document.

Example

This example selects the next occurrence of the word "library" in the selection, regardless of the case.

```
With Selection.Find
    .ClearFormatting
    .MatchWholeWord = True
    .MatchCase = False
    .Execute FindText:="library"
End With
```



MatchControl Property

-
True if find operations match text with matching bidirectional control characters in a right-to-left language document. Read/write **Boolean**.

expression.**MatchControl**

expression Required. An expression that returns a [Find](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the current find operation to match bidirectional control characters.

```
Selection.Find.MatchControl = True
```



MatchDiacritics Property

True if find operations match text with matching diacritics in a right-to-left language document. Read/write **Boolean**.

expression.**MatchDiacritics**

expression Required. An expression that returns a [Find](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the current find operation to match diacritics.

```
Selection.Find.MatchDiacritics = True
```



MatchFuzzy Property

-
True if Microsoft Word uses the nonspecific search options for Japanese text during a search. Read/write **Boolean**.

expression.**MatchFuzzy**

expression Required. An expression that returns a [Find](#) object.

Example

This example conducts a nonspecific search for "ピアノ" in the selected range and selects the next occurrence (for example, "ピヤノ").

```
With Selection.Find
    .ClearFormatting
    .Text = "ピアノ"
    .MatchFuzzy = True
    .Execute Format:=False, Forward:=True, Wrap:=wdFindContinue
End With
```



MatchFuzzyAY Property

True if Microsoft Word ignores the distinction between "7" and "7" following ı-row and ı-row characters during a search. Read/write **Boolean**.

expression.**MatchFuzzyAY**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "ア" and "ヤ" following 1-row and 2-row characters during a search.

`Options.MatchFuzzyAY = True`



MatchFuzzyBV Property

True if Microsoft Word ignores the distinction between "バ" and "ヴァ" and between "ハ" and "ファ" during a search. Read/write **Boolean**.

expression.**MatchFuzzyBV**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "バ" and "ヴ" and between "ハ" and "フ" during a search.

`Options.MatchFuzzyBV = True`



MatchFuzzyByte Property

-
True if Microsoft Word ignores the distinction between full-width and half-width characters (Latin or Japanese) during a search. Read/write **Boolean**.

expression.**MatchFuzzyByte**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between full-width and half-width characters (Latin or Japanese) during a search.

```
Options.MatchFuzzyByte = True
```



MatchFuzzyCase Property

True if Microsoft Word ignores the distinction between uppercase and lowercase letters during a search. Read/write **Boolean**.

expression.**MatchFuzzyCase**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between uppercase and lowercase letters during a search.

```
Options.MatchFuzzyCase = True
```



MatchFuzzyDash Property

True if Microsoft Word ignores the distinction between minus signs, long vowel sounds, and dashes during a search. Read/write **Boolean**.

expression.**MatchFuzzyDash**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between minus signs, long vowel sounds, and dashes during a search.

`Options.MatchFuzzyDash = True`



MatchFuzzyDZ Property

True if Microsoft Word ignores the distinction between "ヂ" and "ジ" and between "ヅ" and "ズ" during a search. Read/write **Boolean**.

expression.**MatchFuzzyDZ**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "ヂ" and "ジ" and between "ヅ" and "ズ" during a search.

`Options.MatchFuzzyDZ = True`



MatchFuzzyHF Property

-
True if Microsoft Word ignores the distinction between "ヒュ" and "フュ" and between "ビュ" and "ヴュ" during a search. Read/write **Boolean**.

expression.**MatchFuzzyHF**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "ヒュ" and "フュ" and between "ビュ" and "ヴュ" during a search.

`Options.MatchFuzzyHF = True`



MatchFuzzyHiragana Property

-
True if Microsoft Word ignores the distinction between hiragana and katakana during a search. Read/write **Boolean**.

expression.**MatchFuzzyHiragana**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between hiragana and katakana during a search.

`Options.MatchFuzzyHiragana = True`



MatchFuzzyIterationMark Property

-
True if Microsoft Word ignores the distinction between types of repetition marks during a search. Read/write **Boolean**.

expression.**MatchFuzzyIterationMark**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between types of repetition marks during a search.

```
Options.MatchFuzzyIterationMark = True
```



MatchFuzzyKanji Property

True if Microsoft Word ignores the distinction between standard and nonstandard kanji ideography during a search. Read/write **Boolean**.

expression.**MatchFuzzyKanji**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between standard and nonstandard Kanji ideography during a search.

`Options.MatchFuzzyKanji = True`



MatchFuzzyKiKu Property

True if Microsoft Word ignores the distinction between "キ" and "ク" before サ-row characters during a search. Read/write **Boolean**.

expression.**MatchFuzzyKiKu**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "キ" and "ク" before サ-row characters during a search.

`Options.MatchFuzzyKiKu = True`



MatchFuzzyOldKana Property

-
True if Microsoft Word ignores the distinction between new kana and old kana characters during a search. Read/write **Boolean**.

expression.**MatchFuzzyOldKana**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between new kana and old kana characters during a search.

```
Options.MatchFuzzyOldKana = True
```



MatchFuzzyProlongedSoundMark Property

True if Microsoft Word ignores the distinction between short and long vowel sounds during a search. Read/write **Boolean**.

expression.**MatchFuzzyProlongedSoundMark**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between short and long vowel sounds during a search.

```
Options.MatchFuzzyProlongedSoundMark = True
```



MatchFuzzyPunctuation Property

True if Microsoft Word ignores the distinction between types of punctuation marks during a search. Read/write **Boolean**.

expression.**MatchFuzzyPunctuation**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between types of punctuation marks during a search

`Options.MatchFuzzyPunctuation = True`



MatchFuzzySmallKana Property

-
True if Microsoft Word ignores the distinction between diphthongs and double consonants during a search. Read/write **Boolean**.

expression.**MatchFuzzySmallKana**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between diphthongs and double consonants during a search.

`Options.MatchFuzzySmallKana = True`



MatchFuzzySpace Property

-
True if Microsoft Word ignores the distinction between space markers used during a search. Read/write **Boolean**.

expression.**MatchFuzzySpace**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between space markers used during a search.

`Options.MatchFuzzySpace = True`



MatchFuzzyTC Property

True if Microsoft Word ignores the distinction between "ツィ", "ティ", and "チ", and between "ディ" and "ヂ" during a search. Read/write **Boolean**.

expression.**MatchFuzzyTC**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "ツイ", "テイ", and "チ", and between "ディ" and "ジ" during a search.

`Options.MatchFuzzyTC = True`



MatchFuzzyZJ Property

True if Microsoft Word ignores the distinction between "ゼ" and "シエ" and between "ゼ" and "ジエ" during a search. Read/write **Boolean**.

expression.**MatchFuzzyZJ**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to ignore the distinction between "セ" and "シ
エ" and between "ゼ" and "ジエ" during a search.

`Options.MatchFuzzyZJ = True`



MatchKashida Property

-
True if find operations match text with matching kashidas in an Arabic language document. Read/write **Boolean**.

expression.**MatchKashida**

expression Required. An expression that returns a [Find](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the current find operation to match kashidas.

```
Selection.Find.MatchKashida = True
```



MatchSoundsLike Property

-
True if words that sound similar to the text to find are returned by the find operation. Read/write **Boolean**.

expression.**MatchSoundsLike**

expression Required. An expression that returns a [Find](#) object.

Remarks

Use the [Text](#) property of the [Find](#) object or the *FindText* argument with the [Execute](#) method to specify the text to be located in a document.

Example

This example selects the next word that sounds like the word "fun" (for instance, "funny") in the selection.

```
With Selection.Find
    .ClearFormatting
    .Text = "fun"
    .MatchFuzzy = False
    .MatchSoundsLike = True
    .Execute Format:=False, Forward:=True, Wrap:=wdFindContinue
End With
```



MatchWholeWord Property

True if the find operation locates only entire words and not text that's part of a larger word. Read/write **Boolean**.

expression.**MatchWholeWord**

expression Required. An expression that returns a [Find](#) object.

Remarks

Use the [Text](#) property of the [Find](#) object or the *FindText* argument with the [Execute](#) method to specify the text to be located in a document.

Example

This example clears all formatting from the find and replace criteria before replacing the word "Inc." with "incorporated" throughout the active document.

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWholeWord = True
    .Execute FindText:="Inc.", _
        ReplaceWith:="incorporated", Replace:=wdReplaceAll
End With
```



MatchWildcards Property

-
True if the text to find contains wildcards. Corresponds to the **Use wildcards** check box in the **Find and Replace** dialog box (**Edit** menu). Read/write **Boolean**.

expression.**MatchWildcards**

expression Required. An expression that returns a [Find](#) object.

Remarks

Use the [Text](#) property of the [Find](#) object or use the *FindText* argument with the [Execute](#) method to specify the text to be located in a document.

Example

This example finds and selects the next three-letter word that begins with "s" and ends with "t."

```
With Selection.Find
    .ClearFormatting
    .Text = "s?t"
    .MatchAllWordForms = False
    .MatchSoundsLike = False
    .MatchFuzzy = False
    .MatchWildcards = True
    .Execute Format:=False, Forward:=True
End With
```



MathCoprocessorAvailable Property

-
True if a math coprocessor is installed and available to Microsoft Word. Read-only **Boolean**.

expression.**MathCoprocessorAvailable**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays a message indicating whether a math coprocessor is installed and available to Word.

```
If Application.MathCoprocessorAvailable = True Then
    MsgBox "A math coprocessor is available."
Else
    MsgBox "A math coprocessor is not installed."
End If
```



MathCoprocesorInstalled Property

True if a math coprocessor is installed on the system. Read-only **Boolean**.

expression.**MathCoprocesorInstalled**

expression Required. An expression that returns a [System](#) object.

Example

This example displays a message if a math coprocessor is installed on the system.

```
If System.MathCoprocessorInstalled = True Then  
    MsgBox "A math coprocessor is installed."  
End If
```



↳ [Show All](#)

Maximum Property

▶ [Maximum property as it applies to the **RecentFiles** object.](#)

Returns or sets the maximum number of recently used files that can appear on the **File** menu. Can be a number from 0 (zero) through 9. Read/write **Long**.

expression.**Maximum**

expression Required. An expression that returns a [RecentFiles](#) object.

▶ [Maximum property as it applies to the **Dictionaries** and **HangulHanjaConversionDictionaries** objects.](#)

Returns the maximum number of custom or conversion dictionaries allowed. Read-only **Long**.

expression.**Maximum**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **RecentFiles** object.](#)

This example disables the list of most recently used files.

```
RecentFiles.Maximum = 0
```

This example increases the number of items on the list of most recently used files by 1.

```
num = RecentFiles.Maximum  
If num <> 9 Then RecentFiles.Maximum = num + 1
```

▶ [As it applies to the **Dictionaries** and **HangulHanjaConversionDictionaries** objects.](#)

This example displays a message if the number of custom dictionaries is equal to the maximum number allowed. If the maximum number hasn't been reached, a custom dictionary named "MyDictionary.dic" is added.

```
If CustomDictionaries.Count = CustomDictionaries.Maximum Then  
    MsgBox "Cannot add another dictionary file"  
Else  
    CustomDictionaries.Add "MyDictionary.dic"  
End If
```



MeaningCount Property

-

Returns the number of entries in the list of meanings found in the thesaurus for the word or phrase. Returns 0 (zero) if no meanings were found. Read-only **Long**.

expression.**MeaningCount**

expression Required. An expression that returns a [SystemInfo](#) object.

Remarks

Each meaning represents a unique list of synonyms for the word or phrase.

The lists of related words, related expressions, and antonyms aren't counted as entries in the list of meanings.

Example

This example checks to see whether any meanings were found for the selection. If any were found, the list of meanings is displayed in the **Immediate** window of the Visual Basic Editor.

```
Set mySynInfo = Selection.Range.SynonymInfo
If mySynInfo.MeaningCount <> 0 Then
    myList = mySynInfo.MeaningList
    For i = 1 To Ubound(myList)
        Debug.Print myList(i)
    Next i
Else
    MsgBox "There were no meanings found."
End If
```



MeaningList Property

Returns the list of meanings for the word or phrase. The list is returned as an array of strings. Read-only **Variant**.

expression.**MeaningList**

expression Required. An expression that returns a [SystemInfo](#) object.

Remarks

The lists of related words, related expressions, and antonyms aren't counted as entries in the list of meanings.

Example

This example checks to see whether any meanings were found for the third word in MyDoc.doc. If so, the meanings are displayed in a series of message boxes.

```
Set mySyn = Documents("MyDoc.doc").Words(3).SynonymInfo
If mySyn.MeaningCount <> 0 Then
    myList = mySyn.MeaningList
    For i = 1 To UBound(myList)
        MsgBox myList(i)
    Next i
Else
    MsgBox "There were no meanings found."
End If
```



↳ [Show All](#)

MeasurementUnit Property

Returns or sets the standard measurement unit for Microsoft Word. Read/write [WdMeasurementUnits](#).

WdMeasurementUnits can be one of these WdMeasurementUnits constants.

wdCentimeters

wdInches

wdMillimeters

wdPicas

wdPoints

expression.**MeasurementUnit**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets the standard measurement unit for Word to points.

```
Options.MeasurementUnit = wdPoints
```

This example returns the current measurement unit selected on the **General** tab in the **Options** dialog box (**Tools** menu).

```
CurrUnit = Options.MeasurementUnit
```



Message Property

Returns or sets the message text for the specified routing slip. The text is used as the body text of the mail message for routing the document. Read/write **String**.

expression.**Message**

expression Required. An expression that returns a [RoutingSlip](#) object.

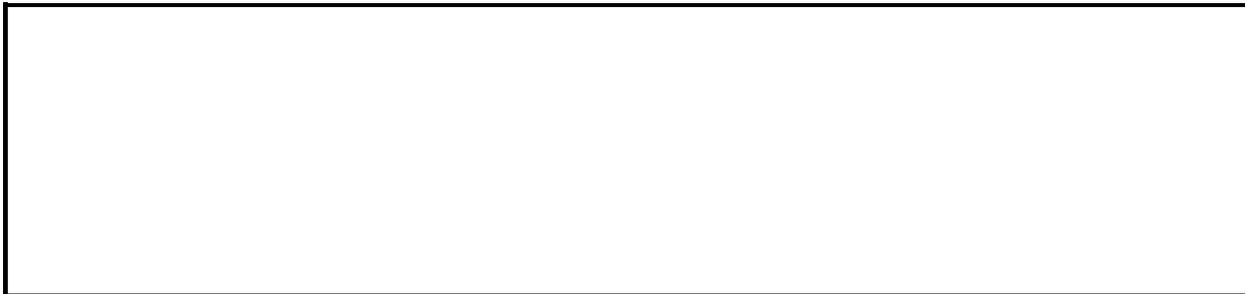
Example

This example adds a routing slip to the active document, sets the subject and message text, adds a recipient, and then routes the document.

```
ActiveDocument.HasRoutingSlip = True
With ActiveDocument.RoutingSlip
    .Subject = "Status Doc"
    .Message = "Please fill in your status."
    .AddRecipient Recipient:="Kate Dresen"
End With
ActiveDocument.Route
```

If "Monthly Report.doc" has a routing slip attached to it, this example displays the message text.

```
Set myDoc = Documents("Monthly Report.doc")
If myDoc.HasRoutingSlip = True _
    Then MsgBox myDoc.RoutingSlip.Message
```



MinimumFontSize Property

Returns or sets the minimum font size (in points) displayed for the specified pane. Read/write **Long**.

expression.**MinimumFontSize**

expression Required. An expression that returns a [Pane](#) object.

Remarks

This property only affects the text as shown in Web layout view. The point sizes that are displayed on the **Formatting** toolbar and used for printing aren't changed.

Example

This example sets the active window to online view and then sets the minimum font size for the active pane to 12 points.

```
With ActiveDocument.ActiveWindow  
    .View.Type = wdWebView  
    .ActivePane.MinimumFontSize = 12  
End With
```

This example returns the minimum font size for the active pane.

```
Msgbox _  
    ActiveDocument.ActiveWindow.ActivePane.MinimumFontSize
```



MirrorMargins Property

True if the inside and outside margins of facing pages are the same width. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

expression.**MirrorMargins**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

If the **MirrorMargins** property is set to **True**, the [LeftMargin](#) property controls the setting for inside margins and the [RightMargin](#) property controls the setting for outside margins.

Example

This example sets the inside margins of the active document to 1 inch (72 points) and the outside margins to 0.5 inch. The [InchesToPoints](#) method is used to convert inches to points.

```
With ActiveDocument.PageSetup
    .MirrorMargins = True
    .LeftMargin = 72
    .RightMargin = InchesToPoints(0.5)
End With
```



Modified Property

-
True if the specified list template is not the built-in list template for that position in the list gallery. Read-only **Boolean**.

expression.**Modified**(*Index*)

expression Required. An expression that returns a [ListGallery](#) object.

Index Required **Long**. A number from 1 to 7 that corresponds to the position of the template in the **Bullets and Numbering** dialog box (**Format** menu). Excluding the **None** option, the templates are numbered from left to right, starting with the top row.

Remarks

Use the [Reset](#) method to set a list template in a list gallery back to the built-in list template.

Example

This example checks to see whether the first template on the **Bulleted** tab in the **Bullets and Numbering** dialog box has been changed. If it has, the list template is reset.

```
temp = ListGalleries(wdBulletGallery).Modified(1)
If temp = True Then
    ListGalleries(wdBulletGallery).Reset(1)
Else
    MsgBox "This is the built-in list template."
End If
```



↳ [Show All](#)

MonthNames Property

Returns or sets the direction for conversion between Hangul and Hanja.
Read/write [WdMonthNames](#).

WdMonthNames can be one of these WdMonthNames constants.

wdMonthNamesEnglish

wdMonthNamesArabic

wdMonthNamesFrench

expression.**MonthNames**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Microsoft Word to convert from Hangul to Hanja by default.

```
Options.MultipleWordConversionsMode = wdHangulToHanja
```



MouseAvailable Property

True if there's a mouse available for the system. Read-only **Boolean**.

expression.**MouseAvailable**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays a message no mouse is available.

```
If Application.MouseAvailable = False Then
    MsgBox "Make sure your mouse is plugged in."
Else
    MsgBox "Mouse is available"
End If
```



↳ [Show All](#)

MultipleWordConversionsMode Property

Returns or sets the direction for conversion between Hangul and Hanja.
Read/write [WdMultipleWordConversionsMode](#).

WdMultipleWordConversionsMode can be one of these
WdMultipleWordConversionsMode constants.

wdHangulToHanja

wdHanjaToHangul

expression.**MultipleWordConversionsMode**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Microsoft Word to convert from Hangul to Hanja by default.

```
Options.MultipleWordConversionsMode = wdHangulToHanja
```



Name Property

Returns or sets the name of the specified object.

Read/write **String** for the following objects: **AutoCorrectEntry**, **AutoTextEntry**, **ColorFormat**, **CustomLabel**, **EmailSignatureEntry**, **Font**, **FormField**, **ListEntry**, **ListTemplate**, **Shape**, **ShapeRange**, and **TableOfAuthoritiesCategory**; read-only **String** for all other objects in the Applies To list.

expression.**Name**

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

Example

This example adds a document variable to the active document and then displays the name of the first document variable.

```
ActiveDocument.Variables.Add Name:="Temp", Value:="1"  
MsgBox ActiveDocument.Variables(1).Name
```

This example returns the name of the first bookmark in Hello.doc.

```
abook = Documents("Hello.doc").Bookmarks(1).Name
```

This example displays the names of the form fields in the active document.

```
If ActiveDocument.FormFields.Count >= 1 Then  
    For Each FF In ActiveDocument.FormFields  
        FFNames = FFNames & FF.Name & vbCrLf  
    Next FF  
    MsgBox FFNames  
End If
```

This example formats the selection as Arial bold.

```
With Selection.Font  
    .Name = "Arial"  
    .Bold = True  
End With
```

This example sets the name of the first list template used in the active document to "myList." A LISTNUM field (linked to the myList template) is then added at the insertion point. The field adopts the formatting of the myList template.

```
If ActiveDocument.ListTemplates.Count >= 1 Then  
    ActiveDocument.ListTemplates(1).Name = "myList"  
    Selection.Collapse Direction:=wdCollapseEnd  
    ActiveDocument.Fields.Add Range:=Selection.Range, _  
        Type:=wdFieldListNum, Text:="myList"  
End If
```



NameAscii Property

Returns or sets the font used for Latin text (characters with character codes from 0 (zero) through 127). Read/write **String**.

expression.NameAscii

expression Required. An expression that returns a [Font](#) object.

Remarks

In the U.S. English version of Microsoft Word, the default value of this property is Times New Roman. Use the [Name](#) property to change the font that's applied to all text and that appears in the **Font** box on the **Formatting** toolbar.

Example

This example sets the font used for Latin text.

```
Selection.Font.NameAscii = "Century"
```



NameBi Property

Returns or sets the name of the font in a right-to-left language document.
Read/write **String**.

expression.**NameBi**

expression Required. An expression that returns a [Font](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example formats the selection with Arial font.

```
With Selection.Font  
    .NameBi = "Arial"  
End With
```



NameFarEast Property

Returns or sets an East Asian font name. Read/write **String**.

expression.NameFarEast

expression Required. An expression that returns a [Font](#) object.

Remarks

In the U.S. English version of Microsoft Word, the default value of this property is Times New Roman. This is the recommended way to return or set the font for Asian text in a document created in an Asian version of Word.

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example displays the East Asian font name that's applied to the selection.

```
MsgBox Selection.Font.NameFarEast
```



↳ [Show All](#)

NameLocal Property

▸ [NameLocal property as it applies to the **Language** object.](#)

Returns the name of a proofing tool language in the language of the user. Read-only **String**.

expression.**NameLocal**

expression Required. An expression that returns a [Language](#) object.

▸ [NameLocal property as it applies to the **Style** object.](#)

Returns the name of a built-in style in the language of the user. Setting this property renames a user-defined style or adds an alias to a built-in style. Read/write **String**.

expression.**NameLocal**

expression Required. An expression that returns a [Style](#) object.

Example

▶ [As it applies to the **Language** object.](#)

This example displays the name of the German language two different ways — first in the language of the user, and then in German.

```
MsgBox Languages(wdGerman).NameLocal  
MsgBox Languages(wdGerman).Name
```

▶ [As it applies to the **Style** object.](#)

This example displays the style name (in the language of the user) applied to the selected paragraphs. If more than one style has been applied to the selection, the first style name is displayed.

```
MsgBox Selection.Paragraphs.Style.NameLocal
```

This example adds the name "MyH1" as the alias for the Heading 1 style in the active document.

```
ActiveDocument.Styles("Heading 1").NameLocal = "MyH1"
```

This example renames the style named "Test" to "Intro."

```
ActiveDocument.Styles("Test").NameLocal = "Intro"
```



NameOther Property

-
Returns or sets the font used for characters with character codes from 128 through 255. Read/write **String**.

expression.**NameOther**

expression Required. An expression that returns a [Font](#) object.

Remarks

In the U.S. English version of Microsoft Word, the default value of this property is Times New Roman. Use the [Name](#) property to change the font that's applied to all text and that appears in the **Font** box on the **Formatting** toolbar.

Example

This example sets the font used for characters with character codes from 128 through 255.

```
Selection.Font.NameOther = "Century"
```



NestingLevel Property

Returns the nesting level of the specified cells, columns, rows, or tables. Read-only **Long**.

expression.**NestingLevel**

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

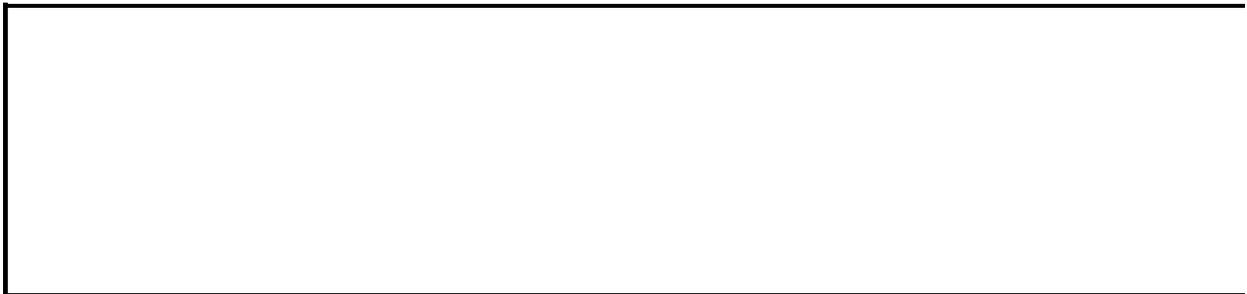
Remarks

The outermost table has a nesting level of 1. The nesting level of each successively nested table is one higher than the previous table.

Example

This example creates a new document, creates a nested table with three levels, and then fills the first cell of each table with its nesting level.

```
Documents.Add
ActiveDocument.Tables.Add Selection.Range, _
    3, 3, wdWord9TableBehavior, wdAutoFitContent
With ActiveDocument.Tables(1).Range
    .Copy
    .Cells(1).Range.Text = .Cells(1).NestingLevel
    .Cells(5).Range.PasteAsNestedTable
With .Cells(5).Tables(1).Range
    .Cells(1).Range.Text = .Cells(1).NestingLevel
    .Cells(5).Range.PasteAsNestedTable
With .Cells(5).Tables(1).Range
    .Cells(1).Range.Text = _
        .Cells(1).NestingLevel
    End With
End With
End With
```



NewColorOnReply Property

True specifies whether a user needs to choose a new color for reply text when replying to e-mail. Read/write **Boolean**.

expression.**NewColorOnReply**

expression Required. An expression that returns an [EmailOptions](#) object.

Remarks

Use the **NewColorOnReply** property if you want the reply text of e-mail messages sent from Microsoft Word to be a different color than the original message.

Example

This example checks to see if a user needs to choose a new color for e-mail reply text and, if not, sets the reply font color to blue.

```
Sub NewColor()  
    With Application.EmailOptions  
        If .NewColorOnReply = False Then  
            .ReplyStyle.Font.Color = wdColorBlue  
        End If  
    End With  
End Sub
```



NewDocument Property

Returns a **NewFile** object that represents a document listed on the **New Document** task pane.

expression.**NewDocument**

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

Example

This example creates a document list item on the **New Document** task pane in the **New From Existing File** section.

```
Sub CreateNewDocument()  
    Application.NewDocument.Add FileName:="C:\NewFile.doc", _  
        Section:=msoNewfromExistingFile, DisplayName:="New File", _  
        Action:=msoCreateNewFile  
End Sub
```



NewMessageSignature Property

Returns or sets the signature that Microsoft Word appends to new e-mail messages. Read/write **String**.

expression.**NewMessageSignature**

expression Required. An expression that returns an [Email](#) object.

Remarks

When setting this property, you must use the name of an e-mail signature that you have created in the **E-mail Options** dialog box, available from the **General** tab of the **Options** dialog box (**Tools** menu).

Example

This example changes the signature Word appends to new outgoing e-mail messages.

```
With Application.EmailOptions.EmailSignature  
    .NewMessageSignature = "Signature1"  
End With
```



Next Property

-

Returns the next object in the collection. Read-only.

Example

This example activates the next window.

```
If Windows.Count > 1 Then ActiveDocument.ActiveWindow.Next.Activate
```

If the selection is in a table, this example selects the contents of the next table cell.

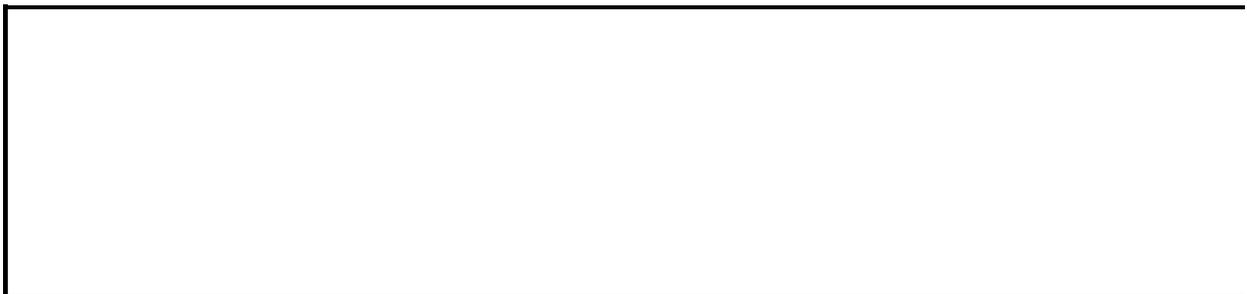
```
If Selection.Information(wdWithInTable) = True Then  
    Selection.Cells(1).Next.Select  
End If
```

This example updates the fields in the first section in the active document as long as the **Next** method returns a **Field** object and the field isn't a FILLIN field.

```
If ActiveDocument.Sections(1).Range.Fields.Count >= 1 Then  
    Set myField = ActiveDocument.Fields(1)  
    While Not (myField Is Nothing)  
        If myField.Type <> wdFieldFillIn Then myField.Update  
        Set myField = myField.Next  
    Wend  
End If
```

This example indents the second paragraph in the selection.

```
Selection.Paragraphs(1).Next.Indent
```



NextParagraphStyle Property

Returns or sets the style to be applied automatically to a new paragraph inserted after a paragraph formatted with the specified style. To set this property, specify either the local name of the next style, an integer or a **WdBuiltinStyle** constant, or an object that represents the next style. Read/write **Variant**.

For a list of the **WdBuiltinStyle** constants, see the [Style](#) property.

expression.**NextParagraphStyle**

expression Required. An expression that returns a [Style](#) object.

Example

This example sets the Heading 1 style to be followed by the Heading 2 style in the active document.

```
ActiveDocument.Styles(wdStyleHeading1).NextParagraphStyle = _  
    ActiveDocument.Styles(wdStyleHeading2)
```

This example creates a new document and adds a paragraph style named "MyStyle." The new style is based on the Normal style, is followed by the Heading 3 style, has a left indent of 1 inch (72 points), and is formatted as bold.

```
Set myDoc = Documents.Add  
Set myStyle = myDoc.Styles.Add(Name:= "MyStyle")  
    With myStyle  
        .BaseStyle = wdStyleNormal  
        .NextParagraphStyle = wdStyleHeading3  
        .ParagraphFormat.LeftIndent = 72  
        .Font.Bold = True  
    End With
```



↳ [Show All](#)

NextStoryRange Property

Returns a [Range](#) object that refers to the next [story](#), as shown in the following table.

Story type	Item returned by the NextStoryRange method
wdMainTextStory, wdFootnotesStory, wdEndnotesStory, and wdCommentsStory	Always returns Nothing
wdTextFrameStory	The story of the next set of linked text boxes
wdEvenPagesHeaderStory, wdPrimaryHeaderStory, wdEvenPagesFooterStory, wdPrimaryFooterStory, wdFirstPageHeaderStory, wdFirstPageFooterStory	The next section's story of the same type

expression.NextStoryRange

expression Required. An expression that returns a [Range](#) object.

Example

This example adds text to the even headers in the first two sections of the active document.

```
If ActiveDocument.Sections.Count >= 2 Then
  With ActiveDocument
    .PageSetup.OddAndEvenPagesHeaderFooter = True
    .Sections(1).Headers(wdHeaderFooterEvenPages) _
      .Range.Text = "Even Header 1"
    .Sections(2).Headers(wdHeaderFooterEvenPages) _
      .LinkToPrevious = False
    .StoryRanges(wdEvenPagesHeaderStory) _
      .NextStoryRange.Text = "Even Header 2"
  End With
End If
```

This example searches each story in the active document for the text "Microsoft Word." The example also applies italic formatting to any instances of this text that it finds.

```
For Each myStoryRange In ActiveDocument.StoryRanges
  myStoryRange.Find.Execute _
    FindText:="Microsoft Word", Forward:=True
  While myStoryRange.Find.Found
    myStoryRange.Italic = True
    myStoryRange.Find.Execute _
      FindText:="Microsoft Word", Forward:=True
  Wend
  While Not (myStoryRange.NextStoryRange Is Nothing)
    Set myStoryRange = myStoryRange.NextStoryRange
    myStoryRange.Find.Execute _
      FindText:="Microsoft Word", Forward:=True
    While myStoryRange.Find.Found
      myStoryRange.Italic = True
      myStoryRange.Find.Execute _
        FindText:="Microsoft Word", Forward:=True
    Wend
  Wend
Next myStoryRange
```



↳ [Show All](#)

Nodes Property

▸ [Nodes property as it applies to the **Diagram** object.](#)

Returns a **DiagramNodes** object that represents the nodes in a diagram.

expression.**Nodes**

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

▸ [Nodes property as it applies to the **Shape** and **ShapeRange** objects.](#)

Returns a **ShapeNodes** collection that represents the geometric description of the specified shape.

expression.**Nodes**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Diagram** object.](#)

This example assumes the first shape in the active document is a diagram, selects the first node, and deletes it.

```
Sub FillDiagramNode()  
    ActiveDocument.Shapes(1).Diagram.Nodes.Item(1).Delete  
End Sub
```

▶ [As it applies to the **Shape** object.](#)

This example adds a smooth node with a curved segment after node four in shape three in the active document. Shape three must be a freeform drawing with at least four nodes.

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



NoLineBreakAfter Property

-

Returns or sets the kinsoku characters after which Microsoft Word will not break a line. Read/write **String**.

Example

This example sets "\$", "(", "[", "\", and "{" as the kinsoku characters after which Microsoft Word will not break a line in the active document.

```
ActiveDocument.NoLineBreakAfter = "$([\{"
```



NoLineBreakBefore Property

-

Returns or sets the kinsoku characters before which Microsoft Word will not break a line. Read/write **String**.

Example

This example sets "!", ")", and "]" as the kinsoku characters before which Microsoft Word will not break a line in the active document.

```
ActiveDocument.NoLineBreakBefore = "!)]"
```



NoLineNumber Property

-
True if line numbers are repressed for the specified paragraphs. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

Remarks

Use the **LineNumbering** property to set line numbers.

Example

This example enables line numbering for the active document. The starting number is set to 1, and the numbering is continuous throughout all sections in the document. Line numbering is then repressed for the second paragraph.

```
With ActiveDocument.PageSetup.LineNumbering
    .Active = True
    .StartingNumber = 1
    .CountBy = 1
    .RestartMode = wdRestartContinuous
End With
ActiveDocument.Paragraphs(2).NoLineNumber = True
```



NoProofing Property

-
Find or **Replacement** object: **True** if Microsoft Word finds or replaces text that the spelling and grammar checker ignores. Read/write **Long**.

Range or **Selection** object: **True** if the spelling and grammar checker ignores the specified text. Returns **wdUndefined** if the **NoProofing** property is set to **True** for only some of the specified text. Read/write **Long**.

Style object: **True** if the spelling and grammar checker ignores text formatted with this style. Read/write **Long**.

Template object: **True** if the spelling and grammar checker ignores documents based on this template. Read/write **Long**.

Example

This example searches for the string "hi" in text that the spelling and grammar checker ignores.

```
With Selection.Find
    .ClearFormatting
    .Text = "hi"
    .NoProofing = True
    .Execute Forward:=True
End With
```

This example marks the current selection to be ignored by the spelling and grammar checker.

```
Selection.NoProofing = True
```

This example sets the spelling and grammar checker to ignore any text in the active document formatted with the style "Test".

```
ActiveDocument.Styles("Test").NoProofing = True
```



↳ [Show All](#)

NormalizedHeight Property

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

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**NormalizedHeight**

expression Required. An expression that returns a [TextEffectFormat](#) object.

Example

This example adds WordArt that contains the text "Test Effect" to myDocument and gives the new WordArt the name "texteff1." The code then makes all characters in the shape named "texteff1" the same height.

```
Set myDocument = ActiveDocument
myDocument.Shapes.AddTextEffect(PresetTextEffect:=msoTextEffect1, _
    Text:="Test Effect", FontName:="Courier New", _
    FontSize:=44, FontBold:=True, _
    FontItalic:=False, Left:=10, Top:=10).Name = "texteff1"
myDocument.Shapes("texteff1").TextEffect.NormalizedHeight = msoTrue
```



NormalTemplate Property

Returns a [Template](#) object that represents the Normal template.

expression.**NormalTemplate**

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

Example

This example inserts the AutoText entry named "Test" from the Normal template, if this entry is contained in the [AutoTextEntries](#) collection.

```
For Each entry In NormalTemplate.AutoTextEntries
    If entry.Name = "Test" Then entry.Insert Where:=Selection.Range
Next entry
```

This example saves the Normal template if changes have been made to it.

```
If NormalTemplate.Saved = False Then NormalTemplate.Save
```



NoShade Property

True if Microsoft Word draws the specified horizontal line without 3-D shading.
Read/write **Boolean**.

expression.NoShade

expression Required. An expression that returns a [HorizontalLineFormat](#) object.

Remarks

You can only use this property with horizontal lines that are not based on an existing image file.

Example

This example adds a horizontal line without any 3-D shading.

```
Selection.InlineShapes.AddHorizontalLineStandard  
ActiveDocument.InlineShapes(1) _  
    .HorizontalLineFormat.NoShade = True
```



NoSpaceBetweenParagraphsOfSameStyle Property

True for Microsoft Word to remove spacing between paragraphs that are formatted using the same style. Read/write **Boolean**.

expression.NoSpaceBetweenParagraphsOfSameStyle

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

Example

This example removes spacing between paragraphs formatted with the "List 1" style. This example assumes that you have a sequence of paragraphs in the active document formatted with a style named "List 1."

```
Sub NoSpace()  
    ActiveDocument.Styles("List 1") _  
        .NoSpaceBetweenParagraphsOfSameStyle = True  
End Sub
```



NumberAcross Property

Returns or sets the number of custom mailing labels across a page. Read/write **Long**.

expression.**NumberAcross**

expression Required. An expression that returns a [CustomLabel](#) object.

Remarks

If this property is changed to a value that isn't valid for the specified mailing label layout, an error occurs.

Example

This example creates a new custom label named "Dept. Labels" and defines the layout, including the number of labels across the page.

```
Set myLabel = Application.MailingLabel.CustomLabels _  
    .Add(Name:="Dept. Labels", DotMatrix:=False)  
With myLabel  
    .Height = InchesToPoints(0.5)  
    .HorizontalPitch = InchesToPoints(2.06)  
    .NumberAcross = 4  
    .NumberDown = 4  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0.28)  
    .TopMargin = InchesToPoints(0.5)  
    .VerticalPitch = InchesToPoints(2)  
    .Width = InchesToPoints(1.75)  
End With
```



NumberDown Property

Returns or sets the number of custom mailing labels down the length of a page.
Read/write **Long**.

expression.**NumberDown**

expression Required. An expression that returns a [CustomLabel](#) object.

Remarks

If this property is changed to a value that isn't valid for the specified mailing label layout, an error occurs.

Example

This example displays the number of labels across and down the page for the first custom label in the [CustomLabels](#) collection.

```
numAcr = Application.MailingLabel.CustomLabels(1).NumberAcross  
numDwn = Application.MailingLabel.CustomLabels(1).NumberDown  
MsgBox Prompt:= "Number of labels across " & numAcr & vbCr _  
    & "Number of labels down " & numDwn & vbCr , _  
    Title:="Label Page Configuration"
```



NumberFormat Property

Returns or sets the number format for the specified list level. Read/write **String**.

expression.**NumberFormat**

expression Required. An expression that returns a [ListLevel](#) object.

Remarks

The percent sign (%) followed by any number from 1 through 9 represents the number style from the respective list level. For example, if you wanted the format for the first level to be "Article I," "Article II," and so on, the string for the **NumberFormat** property would be "Article %1" and the [NumberStyle](#) property would be set to **wdListNumberStyleUpperCaseRoman**.

If the **NumberStyle** property is set to **wdListNumberStyleBullet**, the string for the **NumberFormat** property can only contain one character.

Example

This example creates a list template that indents each level and formats the level with an Arabic numeral and a period. The new list template is then applied to the selection.

```
Set LT = ActiveDocument.ListTemplates.Add(OutlineNumbered:=True)
For x = 1 To 9
  With LT.ListLevels(x)
    .NumberStyle = wdListNumberStyleArabic
    .NumberPosition = InchesToPoints(0.25 * (x - 1))
    .TextPosition = InchesToPoints(0.25 * x)
    .NumberFormat = "%" & x & "."
  End With
Next x
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=LT
```



↳ [Show All](#)

NumberingRule Property

Returns or sets the way footnotes or endnotes are numbered after page breaks or section breaks. Read/write [WdNumberingRule](#).

WdNumberingRule can be one of these WdNumberingRule constants.

wdRestartContinuous

wdRestartPage Applies to the [Footnotes](#) object only.

wdRestartSection

expression.NumberingRule

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

Example

This example restarts endnote numbering after each section break in the active document.

```
ActiveDocument.Endnotes.NumberingRule = wdRestartSection
```

If the footnote numbering in section one is set to restart after each section break, this example sets the numbering to restart on each page.

```
Set myRange = ActiveDocument.Sections(1).Range  
If myRange.Footnotes.NumberingRule = wdRestartSection Then  
    myRange.Footnotes.NumberingRule = wdRestartPage  
End If
```



NumberOfColumns Property

Sets or returns the number of columns for each page of an index. Read/write
Long.

expression.NumberOfColumns

expression Required. An expression that an [Index](#) object.

Remarks

Specifying 0 (zero) sets the number of columns in the index to the same number as in the document.

Example

This example sets the number of columns in the first index to the same number as in the active document.

```
ActiveDocument.Indexes(1).NumberOfColumns = 0
```

This example sets a two-column format for each index in the active document.

```
For Each myIndex In ActiveDocument.Indexes  
    myIndex.NumberOfColumns = 2  
Next myIndex
```



NumberPosition Property

Returns or sets the position (in points) of the number or bullet for the specified [ListLevel](#) object. Read/write **Single**.

expression.**NumberPosition**

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

Remarks

For each list level, you can set the position of the number or bullet, the position of the tab, and the position of the text that wraps.

Example

This example sets the indentation for all the levels of the third outline-numbered list template. Each list level is indented 0.25 inch (18 points) more than the preceding level.

```
r = 0
For Each lev In ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(3).ListLevels
        lev.Alignment = wdListLevelAlignLeft
        lev.NumberPosition = r
        r = r + 18
Next lev
```

This example sets the indent for the first level of the last numbered list template to 0.5 inch.

```
With ListGalleries(wdNumberGallery).ListTemplates(7).ListLevels(1)
    .Alignment = wdListLevelAlignLeft
    .NumberPosition = InchesToPoints(0.5)
End With
```



↳ [Show All](#)

NumberStyle Property

▶ [NumberStyle property as it applies to the **CaptionLabel** object.](#)

Returns or sets the number style for the [CaptionLabel](#) object. Read/write [WdCaptionNumberStyle](#).

WdCaptionNumberStyle can be one of these WdCaptionNumberStyle constants.

wdCaptionNumberStyleArabicFullWidth

wdCaptionNumberStyleArabicLetter2

wdCaptionNumberStyleGanada

wdCaptionNumberStyleHanjaReadDigit

wdCaptionNumberStyleHebrewLetter2

wdCaptionNumberStyleHindiCardinalText

wdCaptionNumberStyleHindiLetter2

wdCaptionNumberStyleKanjiDigit

wdCaptionNumberStyleArabic

wdCaptionNumberStyleArabicLetter1

wdCaptionNumberStyleChosung

wdCaptionNumberStyleHanjaRead

wdCaptionNumberStyleHebrewLetter1

wdCaptionNumberStyleHindiArabic

wdCaptionNumberStyleHindiLetter1

wdCaptionNumberStyleKanji

wdCaptionNumberStyleKanjiTraditional

wdCaptionNumberStyleLowercaseLetter

wdCaptionNumberStyleLowercaseRoman

wdCaptionNumberStyleNumberInCircle

wdCaptionNumberStyleSimpChinNum2

wdCaptionNumberStyleSimpChinNum3

wdCaptionNumberStyleThaiArabic
wdCaptionNumberStyleThaiCardinalText
wdCaptionNumberStyleThaiLetter
wdCaptionNumberStyleTradChinNum2
wdCaptionNumberStyleTradChinNum3
wdCaptionNumberStyleUppercaseLetter
wdCaptionNumberStyleUppercaseRoman
wdCaptionNumberStyleVietCardinalText
wdCaptionNumberStyleZodiac1
wdCaptionNumberStyleZodiac2

expression.**NumberStyle**

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

► [NumberStyle](#) property as it applies to the [EndnoteOptions](#), [Endnotes](#), [FootnoteOptions](#), and [Footnotes](#) objects.

Returns or sets the number style for the [EndnoteOptions](#), [Endnotes](#), [FootnoteOptions](#), and [Footnotes](#) objects. Read/write [WdNoteNumberStyle](#).

WdNoteNumberStyle can be one of these WdNoteNumberStyle constants.

wdNoteNumberStyleArabic
wdNoteNumberStyleArabicLetter1
wdNoteNumberStyleHanjaRead
wdNoteNumberStyleHebrewLetter1
wdNoteNumberStyleHindiArabic
wdNoteNumberStyleHindiLetter1
wdNoteNumberStyleKanji
wdNoteNumberStyleKanjiTraditional
wdNoteNumberStyleLowercaseRoman
wdNoteNumberStyleSimpChinNum1
wdNoteNumberStyleSymbol
wdNoteNumberStyleThaiCardinalText
wdNoteNumberStyleTradChinNum1

wdNoteNumberStyleUppercaseLetter
wdNoteNumberStyleVietCardinalText
wdNoteNumberStyleArabicFullWidth
wdNoteNumberStyleArabicLetter2
wdNoteNumberStyleHanjaReadDigit
wdNoteNumberStyleHebrewLetter2
wdNoteNumberStyleHindiCardinalText
wdNoteNumberStyleHindiLetter2
wdNoteNumberStyleKanjiDigit
wdNoteNumberStyleLowercaseLetter
wdNoteNumberStyleNumberInCircle
wdNoteNumberStyleSimpChinNum2
wdNoteNumberStyleThaiArabic
wdNoteNumberStyleThaiLetter
wdNoteNumberStyleTradChinNum2
wdNoteNumberStyleUppercaseRoman

expression.**NumberStyle**

expression Required. An expression that returns an **EndnoteOptions**, **Endnotes**, **FootnoteOptions**, or **Footnotes** object.

► [NumberStyle property as it applies to the ListLevel object.](#)

Returns or sets the number style for the [ListLevel](#) object. Read/write [WdListNumberStyle](#).

WdListNumberStyle can be one of these WdListNumberStyle constants.

wdListNumberStyleAiueo
wdListNumberStyleArabic
wdListNumberStyleArabic2
wdListNumberStyleArabicLZ
wdListNumberStyleCardinalText
wdListNumberStyleChosung
wdListNumberStyleGanada

wdListNumberStyleGBNum1
wdListNumberStyleGBNum2
wdListNumberStyleGBNum3
wdListNumberStyleGBNum4
wdListNumberStyleHangul
wdListNumberStyleHanja
wdListNumberStyleHanjaRead
wdListNumberStyleHanjaReadDigit
wdListNumberStyleHebrew1
wdListNumberStyleHebrew2
wdListNumberStyleHindiArabic
wdListNumberStyleHindiCardinalText
wdListNumberStyleHindiLetter1
wdListNumberStyleHindiLetter2
wdListNumberStyleIroha
wdListNumberStyleIrohaHalfWidth
wdListNumberStyleKanji
wdListNumberStyleKanjiDigit
wdListNumberStyleKanjiTraditional
wdListNumberStyleKanjiTraditional2
wdListNumberStyleLegal
wdListNumberStyleLegalLZ
wdListNumberStyleLowercaseLetter
wdListNumberStyleLowercaseRoman
wdListNumberStyleLowercaseRussian
wdListNumberStyleNone
wdListNumberStyleNumberInCircle
wdListNumberStyleOrdinal
wdListNumberStyleOrdinalText
wdListNumberStylePictureBullet
wdListNumberStyleSimpChinNum1
wdListNumberStyleSimpChinNum2
wdListNumberStyleSimpChinNum3

wdListNumberStyleSimpChinNum4
wdListNumberStyleThaiArabic
wdListNumberStyleThaiCardinalText
wdListNumberStyleThaiLetter
wdListNumberStyleTradChinNum1
wdListNumberStyleTradChinNum2
wdListNumberStyleTradChinNum3
wdListNumberStyleTradChinNum4
wdListNumberStyleUppercaseLetter
wdListNumberStyleUppercaseRoman
wdListNumberStyleUppercaseRussian
wdListNumberStyleVietCardinalText
wdListNumberStyleZodiac1
wdListNumberStyleZodiac2
wdListNumberStyleZodiac3
wdListNumberStyleAiueoHalfWidth
wdListNumberStyleArabic1
wdListNumberStyleArabicFullWidth
wdListNumberStyleBullet

expression.**NumberStyle**

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

► [NumberStyle property as it applies to the PageNumbers object.](#)

Returns or sets the number style for the [PageNumbers](#) object. Read/write [WdPageNumberStyle](#).

WdPageNumberStyle can be one of these WdPageNumberStyle constants.

wdPageNumberStyleArabic
wdPageNumberStyleArabicLetter1
wdPageNumberStyleHanjaRead
wdPageNumberStyleHebrewLetter1
wdPageNumberStyleHindiArabic

wdPageNumberStyleHindiLetter1
wdPageNumberStyleKanji
wdPageNumberStyleKanjiTraditional
wdPageNumberStyleLowercaseRoman
wdPageNumberStyleNumberInDash
wdPageNumberStyleSimpChinNum2
wdPageNumberStyleThaiCardinalText
wdPageNumberStyleTradChinNum1
wdPageNumberStyleUppercaseLetter
wdPageNumberStyleVietCardinalText
wdPageNumberStyleArabicFullWidth
wdPageNumberStyleArabicLetter2
wdPageNumberStyleHanjaReadDigit
wdPageNumberStyleHebrewLetter2
wdPageNumberStyleHindiCardinalText
wdPageNumberStyleHindiLetter2
wdPageNumberStyleKanjiDigit
wdPageNumberStyleLowercaseLetter
wdPageNumberStyleNumberInCircle
wdPageNumberStyleSimpChinNum1
wdPageNumberStyleThaiArabic
wdPageNumberStyleThaiLetter
wdPageNumberStyleTradChinNum2
wdPageNumberStyleUppercaseRoman

expression.**NumberStyle**

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

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

▶ [As it applies to the **CaptionLabel** object.](#)

This example inserts a caption at the insertion point. The caption letter is formatted as an uppercase letter.

```
CaptionLabels(wdCaptionFigure).NumberStyle = _  
    wdCaptionNumberStyleUppercaseLetter  
Selection.Collapse Direction:=wdCollapseEnd  
Selection.InsertCaption Label:=wdCaptionFigure
```

▶ [As it applies to the **ListLevel** object.](#)

This example creates an alternating number style for the third outline-numbered list template.

```
Set myTemp = ListGalleries(wdOutlineNumberGallery).ListTemplates(3)  
For i = 1 to 9  
    If i Mod 2 = 0 Then  
        myTemp.ListLevels(i).NumberStyle = _  
            wdListNumberStyleUppercaseRoman  
    Else  
        myTemp.ListLevels(i).NumberStyle = _  
            wdListNumberStyleLowercaseRoman  
    End If  
Next i
```

This example changes the number style to uppercase letters for every outline-numbered list in the active document.

```
For Each lt In ActiveDocument.ListTemplates  
    For Each ll In lt.listlevels  
        ll.NumberStyle = wdListNumberStyleUppercaseLetter  
    Next ll  
Next lt
```

▶ [As it applies to the **Footnote** and **Endnote** options.](#)

This example sets the formatting for footnotes and endnotes in the active document.

```
With ActiveDocument
    .Footnotes.NumberStyle = wdNoteNumberStyleLowercaseRoman
    .Endnotes.NumberStyle = wdNoteNumberStyleUppercaseRoman
End With
```

▶ [As it applies to the **PageNumbers** object.](#)

This example formats the page numbers in the active document's footer as lowercase roman numerals.

```
For Each sec In ActiveDocument.Sections
    sec.Footers(wdHeaderFooterPrimary).PageNumbers _
        .NumberStyle = wdPageNumberStyleLowercaseRoman
Next sec
```



NumLock Property

Returns the state of the NUM LOCK key. **True** if the keys on the numeric keypad insert numbers, **False** if the keys move the insertion point. Read-only **Boolean**.

expression.**NumLock**

expression Required. An expression that returns an [Application](#) object.

Example

This example returns the current state of the NUM LOCK key.

```
theState = Application.NumLock
```



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.

expression.**Object**

expression Required. An expression that returns an [OLEFormat](#) object.

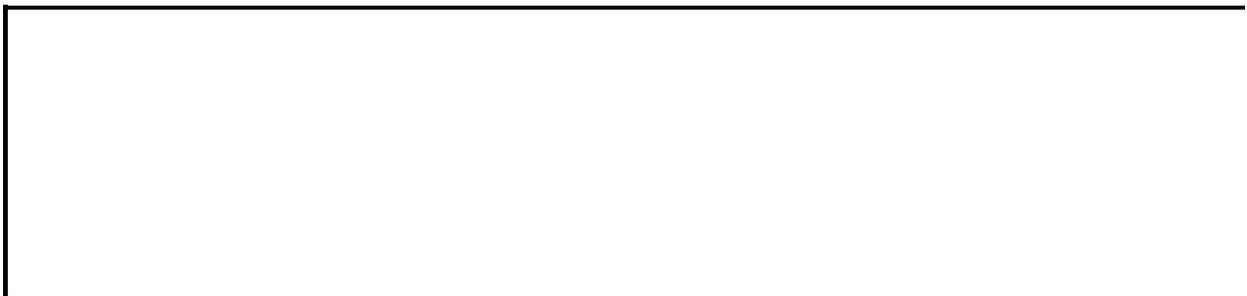
Example

This example sets the value of the first shape on the active document. For the example to work, this first shape must be an ActiveX control (for example, a check box or an option button).

```
With ActiveDocument.Shapes(1).OLEFormat
    .Activate
    Set myObj = .Object
End With
myObj.Value = True
```

This example adds a new ActiveX control to the active document. The example then activates the new option button and sets some of its properties.

```
Set myOB = ActiveDocument.Shapes _
    .AddOLEControl(ClassType:="Forms.OptionButton.1")
With myOB.OLEFormat
    .Activate
    Set myObj = .Object
End With
With myObj
    .Value = False
    .Caption = "My Caption"
    .AutoSize = True
End With
```



↳ [Show All](#)

Obscured Property

-

MsoTrue if the shadow of the specified shape appears filled in and is obscured by the shape, even if the shape has no fill. **MsoFalse** if the shadow has no fill and the outline of the shadow is visible through the shape if the shape has no fill. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.Obscured

expression Required. An expression that returns a [ShadowFormat](#) object.

Example

This example sets the horizontal and vertical offsets for the shadow of shape three on myDocument. 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.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
    .Obscured = msoTrue
End With
```



OddAndEvenPagesHeaderFooter Property

True if the specified [PageSetup](#) object has different headers and footers for odd-numbered and even-numbered pages. Can be **True**, **False**, or **wdUndefined**.
Read/write **Long**.

expression.**OddAndEvenPagesHeaderFooter**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example creates different headers and footers for odd-numbered and even-numbered pages in Document1.

```
Set myDoc = Documents("Document1")
myDoc.PageSetup.OddAndEvenPagesHeaderFooter = True
With myDoc.Sections(1)
    .Headers(wdHeaderFooterPrimary).Range _
        .InsertAfter "Odd Header"
    .Headers(wdHeaderFooterEvenPages).Range _
        .InsertAfter "Even Header"
End With
```



OffsetX Property

Returns or sets the horizontal offset (in points) 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 **Single**.

expression.**OffsetX**

expression Required. An expression that returns a [ShadowFormat](#) object.

Remarks

If you want to nudge a shadow horizontally or vertically from its current position without having to specify an absolute position, use the [IncrementOffsetX](#) or [IncrementOffsetY](#) method.

Example

This example sets the horizontal and vertical offsets for the shadow of shape three on myDocument. 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.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```



OffsetY Property

Returns or sets the vertical offset (in points) 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 **Single**.

expression.**OffsetY**

expression Required. An expression that returns a [ShadowFormat](#) object.

Remarks

If you want to nudge a shadow horizontally or vertically from its current position without having to specify an absolute position, use the [IncrementOffsetX](#) or [IncrementOffsetY](#) method.

Example

This example sets the horizontal and vertical offsets for the shadow of shape three in myDocument. 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.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```



OLEFormat Property

-

Returns an [OLEFormat](#) object that represents the OLE characteristics (other than linking) for the specified shape, inline shape, or field. Read-only.

Example

This example loops through all the floating shapes on the active document and sets all linked Microsoft Excel worksheets to be updated automatically.

```
For Each s In ActiveDocument.Shapes
    If s.Type = msoLinkedOLEObject Then
        If s.OLEFormat.ProgID = "Excel.Sheet" Then
            s.LinkFormat.AutoUpdate = True
        End If
    End If
Next
```



OpenEncoding Property

Returns the encoding used to open the specified document. Read-only [MsoEncoding](#).

MsoEncoding can be one of these MsoEncoding constants; however, you cannot use any of the constants that have the suffix **AutoDetect**. These constants are used by the [ReloadAs](#) method.

msoEncodingOEMMultilingualLatinI

msoEncodingOEMNordic

msoEncodingOEMTurkish

msoEncodingSimplifiedChineseAutoDetect

msoEncodingT61

msoEncodingTaiwanEten

msoEncodingTaiwanTCA

msoEncodingTaiwanWang

msoEncodingTraditionalChineseAutoDetect

msoEncodingTurkish

msoEncodingUnicodeLittleEndian

msoEncodingUTF7

msoEncodingVietnamese

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDIKoreanExtendedAndKorean

msoEncodingEBCDICMultilingualROECELatin2

msoEncodingEBCDICSerbianBulgarian

msoEncodingEBCDICThai

msoEncodingEBCDICTurkishLatin5

msoEncodingEBCDICUSCanada

msoEncodingEBCDICUSCanadaAndTraditionalChinese

msoEncodingOEMModernGreek
msoEncodingOEMMultilingualLatinII
msoEncodingOEMPortuguese
msoEncodingOEMUnitedStates
msoEncodingSimplifiedChineseGBK
msoEncodingTaiwanCNS
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTeleText
msoEncodingThai
msoEncodingTraditionalChineseBig5
msoEncodingUnicodeBigEndian
msoEncodingUSASCII
msoEncodingUTF8
msoEncodingWestern
msoEncodingArabic
msoEncodingArabicASMO
msoEncodingArabicAutoDetect
msoEncodingArabicTransparentASMO
msoEncodingAutoDetect
msoEncodingBaltic
msoEncodingCentralEuropean
msoEncodingCyrillic
msoEncodingCyrillicAutoDetect
msoEncodingEBCDICArabic
msoEncodingEBCDICDenmarkNorway
msoEncodingEBCDICFinlandSweden
msoEncodingEBCDICFrance
msoEncodingEBCDICGermany
msoEncodingEBCDICGreek
msoEncodingEBCDICGreekModern
msoEncodingEBCDICHebrew
msoEncodingEBCDICIcelandic
msoEncodingEBCDICInternational

msoEncodingEBCDICItaly
msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese
msoEncodingEBCDIKKoreanExtended
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICRussian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICTurkish
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
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

expression.**OpenEncoding**

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

Example

This example tests whether the current document was opened with UTF7 encoding.

```
If ActiveDocument.OpenEncoding = msoEncodingUTF7 Then
    MsgBox "This is a UTF7-encoded text file!"
Else
    MsgBox "This is not a UTF7-encoded text file!"
End If
```



↳ [Show All](#)

OpenFormat Property

Returns the file format of the specified file converter. Can be any valid [WdOpenFormat](#) constant, or it can be a unique number that represents an external file converter. Read-only **Long**.

WdOpenFormat can be one of these WdOpenFormat constants.

wdOpenFormatAllWord

wdOpenFormatAuto

wdOpenFormatDocument

wdOpenFormatEncodedText

wdOpenFormatRTF

wdOpenFormatTemplate

wdOpenFormatText

wdOpenFormatUnicodeText

wdOpenFormatWebPages

expression.**OpenFormat**

expression Required. An expression that returns a [FileConverter](#) object.

Example

This example displays the unique format value and the format name for the converters you can use to open documents.

```
For Each fc In FileConverters
    If fc.CanOpen = True Then _
        MsgBox fc.OpenFormat & vbCr & fc.FormatName
Next fc
```

This example opens the file named "Data.wp" by using the WordPerfect 6x file converter.

```
Documents.Open FileName:="C:\Data.wp", _
    Format:=FileConverters("WordPerfect6x").OpenFormat
```



OperatingSystem Property

Returns the name of the current operating system (for example, "Windows" or "Windows NT"). Read-only **String**.

expression.**OperatingSystem**

expression Required. An expression that returns a [System](#) object.

Example

This example displays a message that includes the name of the current operating system.

```
MsgBox "This computer is running " & System.OperatingSystem
```



OptimizeForBrowser Property

-

True if Microsoft Word optimizes new Web pages created in Word for the Web browser specified by the [BrowserLevel](#) property (for the [DefaultWebOptions](#) object). **True** if Word optimizes the specified Web page for the Web browser specified by the [BrowserLevel](#) property (for the [WebOptions](#) object).
Read/write **Boolean**.

expression.**OptimizeForBrowser**

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

Example

This example sets Word to optimize new Web pages for Microsoft Internet Explorer 5 and creates a Web page based on this setting.

```
With Application.DefaultWebOptions
    .BrowserLevel _
        = wdBrowserLevelMicrosoftInternetExplorer5
    .OptimizeForBrowser = True
End With
Documents.Add DocumentType:=wdNewWebPage
```

This example creates a new Web page and optimizes it for Microsoft Internet Explorer 5.

```
Documents.Add DocumentType:=wdNewWebPage
With ActiveDocument.WebOptions
    .BrowserLevel _
        = wdBrowserLevelMicrosoftInternetExplorer5
    .OptimizeForBrowser = True
End With
```



OptimizeForWord97 Property

-
True if Microsoft Word optimizes the current document for viewing in Word 97 by disabling any incompatible formatting. Read/write **Boolean**.

Remarks

To optimize all new documents for Word 97 by default, use the [OptimizeForWord97byDefault](#) property.

Example

This example checks the current document to see if it's optimized for Word 97; if it isn't, the example asks the user whether it should be.

```
If ActiveDocument.OptimizeForWord97 = False Then
    x = MsgBox("Is this document targeted at " _
        & "Word 97 users?", vbYesNo)
    If x = vbYes Then _
        ActiveDocument.OptimizeForWord97 = True
End If
```



OptimizeForWord97byDefault Property

True if Microsoft Word optimizes all new documents for viewing in Word 97 by disabling any incompatible formatting. Read/write **Boolean**.

Remarks

To optimize a single document for Word 97, use the [OptimizeForWord97](#) property.

Example

This example sets Word to disable all formatting in new documents that's incompatible with Word 97, and then it creates a new document whose **OptimizeForWord97** property is automatically set to **True**.

```
Options.OptimizeForWord97byDefault = True  
MsgBox Documents.Add(DocumentType:=wdNewBlankDocument) _  
    .OptimizeForWord97
```



Options Property

Returns an [Options](#) object that represents application settings in Microsoft Word.

expression.**Options**

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

Example

This example disables fast saves and then saves the active document.

```
Options.AllowFastSave = False  
ActiveDocument.Save
```

This example prints Sales.doc with comments and field results.

```
With Options  
    .PrintFieldCodes = False  
    .PrintComments = True  
End With  
Documents("Sales.doc").PrintOut
```



OrganizeInFolder Property

-

True if all supporting files, such as background textures and graphics, are organized in a separate folder when you save the specified document as a Web page. **False** if supporting files are saved in the same folder as the Web page. The default value is **True**. Read/write **Boolean**.

expression.**OrganizeInFolder**

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

Remarks

The new folder is created in the folder where you have saved the Web page and is named after the document. If long file names are used, a suffix is added to the folder name. The [FolderSuffix](#) property returns wither the folder suffix for the language support you have selected or installed or the default folder suffix.

If you save a document that was previously saved with the **OrganizeInFolder** property set to a different value, Microsoft Word automatically moves the supporting files into or out of the folder, as appropriate.

If you don't use long file names (that is, if the [UseLongFileNames](#) property is set to **False**), Microsoft Word automatically saves any supporting files in a separate folder. The files cannot be saved in the same folder as the Web page.

Example

This example specifies that all supporting files are saved in the same folder when the document is saved as a Web page.

```
Application.DefaultWebOptions.OrganizeInFolder = False
```



↳ [Show All](#)

Orientation Property

▶ [Orientation property as it applies to the **PageSetup** object.](#)

Returns or sets the orientation of the page. Read/write [WdOrientation](#).

WdOrientation can be one of these WdOrientation constants.

wdOrientLandscape

wdOrientPortrait

expression.**Orientation**

expression Required. An expression that returns a [PageSetup](#) object.

▶ [Orientation property as it applies to the **Range** and **Selection** objects.](#)

Returns or sets the orientation of text in a range or selection when the Text Direction feature is enabled. Read/write [WdTextOrientation](#).

WdTextOrientation can be one of these WdTextOrientation constants.

wdTextOrientationDownward

wdTextOrientationHorizontal

wdTextOrientationHorizontalRotatedFarEast

wdTextOrientationUpward

wdTextOrientationVerticalFarEast

expression.**Orientation**

expression Required. An expression that returns one of the above objects.

▶ [Orientation property as it applies to the **TextFrame** object.](#)

Returns or sets the orientation of the text inside the frame. Read/write [MsoTextOrientation](#).

MsoTextOrientation can be one of these MsoTextOrientation constants.

msoTextOrientationDownward

msoTextOrientationHorizontal

msoTextOrientationHorizontalRotatedFarEast

msoTextOrientationMixed

msoTextOrientationUpward

msoTextOrientationVertical

msoTextOrientationVerticalFarEast

expression.**Orientation**

expression Required. An expression that returns a [TextFrame](#) object.

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.

You can set the orientation for a text frame or for a range or selection that happens to occur inside a text frame. For information about the difference between a text frame and a text box, see the [TextFrame](#) object.

Example

▶ [As it applies to the **TextFrame** object](#)

This example creates a new document, inserts text into it, uses this text to create a text box, and then sets the orientation of the text frame so that the text slopes upward.

```
Set mydoc = Documents.Add
Selection.TypeText "This is some text."
mydoc.Content.Select
Selection.CreateTextbox
mydoc.Shapes(1).TextFrame.Orientation = msoTextOrientationUpward
```

▶ [As it applies to the **PageSetup** object.](#)

This example changes the orientation of the document named "MyDocument.doc" and then prints the document. The example then changes the orientation of the document back to portrait.

```
Set myDoc = Documents("MyDocument.doc")
With myDoc
    .PageSetup.Orientation = wdOrientLandscape
    .PrintOut
    .PageSetup.Orientation = wdOrientPortrait
End With
```



OtherCorrectionsAutoAdd Property

-

True if Microsoft Word automatically adds words to the list of AutoCorrect exceptions on the **Other Corrections** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect Options** command, **Tools** menu). Word adds a word to this list if you delete and then retype a word that you didn't want Word to correct. Read/write **Boolean**.

expression.**OtherCorrectionsAutoAdd**

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example sets Word to automatically add words to the list of AutoCorrect exceptions.

```
AutoCorrect.OtherCorrectionsAutoAdd = True
```



OtherCorrectionsExceptions Property

Returns an [OtherCorrectionsExceptions](#) collection that represents the list of words that Microsoft Word won't correct automatically. This list corresponds to the list of AutoCorrect exceptions on the **Other Corrections** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect** command, **Tools** menu).

expression.**OtherCorrectionsExceptions**

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example prompts the user to delete or keep each AutoCorrect exception on the **Other Corrections** tab in the **AutoCorrect Exceptions** dialog box.

```
For Each anEntry In _
    AutoCorrect.OtherCorrectionsExceptions
    response = MsgBox("Delete entry: " _
        & anEntry.Name, vbYesNoCancel)
    If response = vbYes Then
        anEntry.Delete
    Else
        If response = vbCancel Then End
    End If
Next anEntry
```



↳ [Show All](#)

OtherPagesTray Property

Returns or sets the paper tray to be used for all but the first page of a document or section. Read/write [WdPaperTray](#).

WdPaperTray can be one of these WdPaperTray constants.

wdPrinterAutomaticSheetFeed

wdPrinterDefaultBin

wdPrinterEnvelopeFeed

wdPrinterFormSource

wdPrinterLargeCapacityBin

wdPrinterLargeFormatBin

wdPrinterLowerBin

wdPrinterManualEnvelopeFeed

wdPrinterManualFeed

wdPrinterMiddleBin

wdPrinterOnlyBin

wdPrinterPaperCassette

wdPrinterSmallFormatBin

wdPrinterTractorFeed

wdPrinterUpperBin

expression.**OtherPagesTray**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example sets the tray to be used for printing all but the first page of each section in the active document.

```
ActiveDocument.PageSetup.OtherPagesTray = wdPrinterUpperBin
```

This example sets the tray to be used for printing all but the first page of each section in the selection.

```
Selection.PageSetup.OtherPagesTray = wdPrinterLowerBin
```



Outline Property

-
True if the font is formatted as outline. Returns **True**, **False**, or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**Outline**

expression Required. An expression that returns a [Font](#) object.

Example

This example applies outline font formatting to the first three words in the active document.

```
Set myRange = ActiveDocument.Range(Start:= _  
    ActiveDocument.Words(1).Start, _  
    End:=ActiveDocument.Words(3).End)  
myRange.Font.Outline = True
```

This example toggles outline formatting for the selected text.

```
Selection.Font.Outline = wdToggle
```

This example removes outline font formatting from the selection if outline formatting is partially applied to the selection.

```
Set myFont = Selection.Font  
If myFont.Outline = wdUndefined Then  
    myFont.Outline = False  
End If
```



OutlineLevel Property

Returns or sets the outline level for the specified paragraphs. Read/write [wdOutlineLevel](#).

Can be one of the following **WdOutlineLevel** constants.

wdOutlineLevel1

wdOutlineLevel2

wdOutlineLevel3

wdOutlineLevel4

wdOutlineLevel5

wdOutlineLevel6

wdOutlineLevel7

wdOutlineLevel8

wdOutlineLevel9

wdOutlineLevelBodyText.

expression.**OutlineLevel**

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

Remarks

If a paragraph has a heading style applied to it (Heading 1 through Heading 9), the outline level is the same as the heading style and cannot be changed.

Outline levels are visible only in outline view or the document map pane.

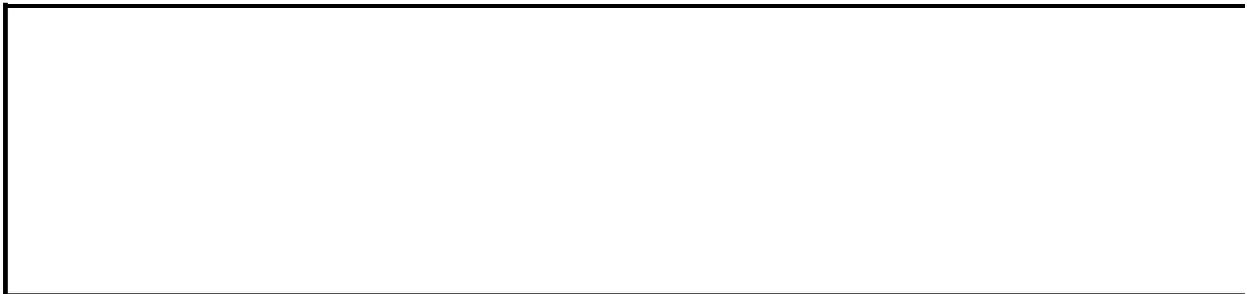
Example

This example returns the outline level of the first paragraph in the active document.

```
temp = ActiveDocument.Paragraphs(1).OutlineLevel
```

This example sets the outline level for each paragraph in the active document. First the Normal style is applied to all paragraphs. The **Mod** operator is used to determine which outline level (1, 2, or 3) to apply to successive paragraphs in the document, and then the view is changed to outline view.

```
Set myParas = ActiveDocument.Paragraphs
ActiveDocument.Paragraphs.Style = wdStyleNormal
For x = 1 To myParas.Count
    If x Mod 3 = 1 Then
        myParas(x).OutlineLevel = wdOutlineLevel1
    ElseIf x Mod 3 = 2 Then
        myParas(x).OutlineLevel = wdOutlineLevel2
    Else
        myParas(x).OutlineLevel = wdOutlineLevel3
    End If
Next x
ActiveDocument.ActiveWindow.View.Type = wdOutlineView
```



OutlineNumbered Property

-
True if the specified [ListTemplate](#) object is outline numbered. Read/write **Boolean**.

expression.**OutlineNumbered**

expression Required. An expression that returns a [ListTemplate](#) object.

Remarks

Setting this property to **False** converts the list template to a single-level list that uses the formatting of the first level.

You cannot set this property for a **ListTemplate** object returned from a [ListGallery](#) object.

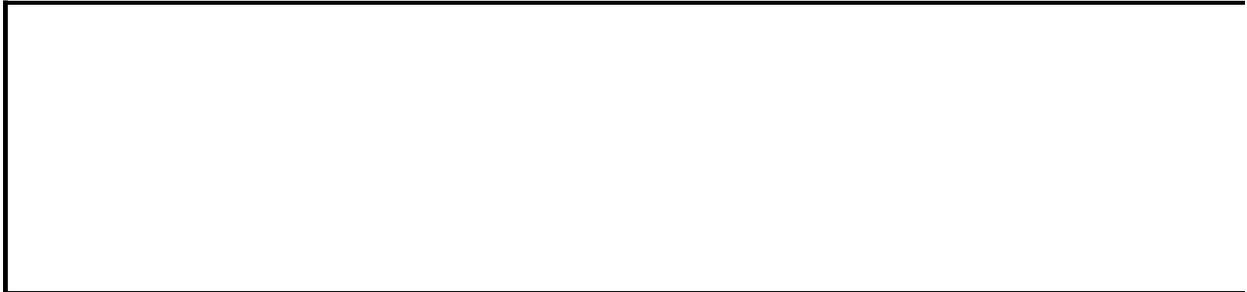
Example

This example changes the selected outline-numbered list to a single-level numbered list.

```
Selection.Range.ListFormat.ListTemplate.OutlineNumbered = False
```

This example checks to see whether the third list in MyDoc.doc is an outline-numbered list. If it is, the third outline-numbered list template is applied to it.

```
Set myltemp = Documents("MyDoc.doc").Lists(3).Range _  
    .ListFormat.ListTemplate  
num = myltemp.OutlineNumbered  
If num = True Then ActiveDocument.Lists(3).ApplyListTemplate _  
    ListTemplate:=ListGalleries(wdOutlineNumberGallery) _  
    .ListTemplates(3)
```



↳ [Show All](#)

OutsideColor Property

Returns or sets the 24-bit color of the outside borders. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic

wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen
wdColorLightTurquoise

wdColorLime
wdColorOrange
wdColorPink

expression.**OutsideColor**

expression Required. An expression that returns a [Borders](#) object.

Remarks

If the [OutsideLineStyle](#) property is set to either `wdLineStyleNone` or `False`, setting this property has no effect.

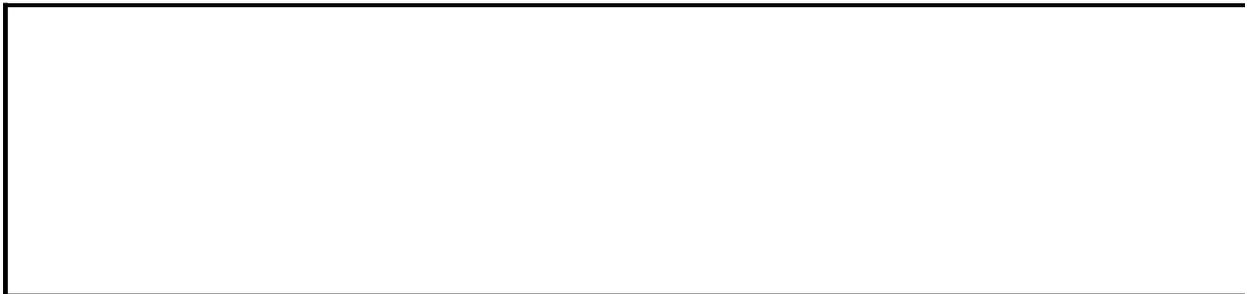
Example

This example adds borders between rows and between columns in the first table of the active document, and then it sets the colors for both the inside and outside borders.

```
If ActiveDocument.Tables.Count >= 1 Then
    Set myTable = ActiveDocument.Tables(1)
    With myTable.Borders
        .InsideLineStyle = True
        .InsideColor = wdColorBrightGreen
        .OutsideColor = wdColorDarkTeal
    End With
End If
```

This example adds a dark red, 0.75-point double border around the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).Borders
    .OutsideLineStyle = wdLineStyleDouble
    .OutsideLineWidth = wdLineWidth075pt
    .OutsideColor = wdColorDarkRed
End With
```



↳ [Show All](#)

OutsideColorIndex Property

Returns or sets the color of the outside borders. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**OutsideColorIndex**

expression Required. An expression that returns a [Borders](#) object.

Remarks

If the [OutsideLineStyle](#) property is set to either `wdLineStyleNone` or `False`, setting this property has no effect.

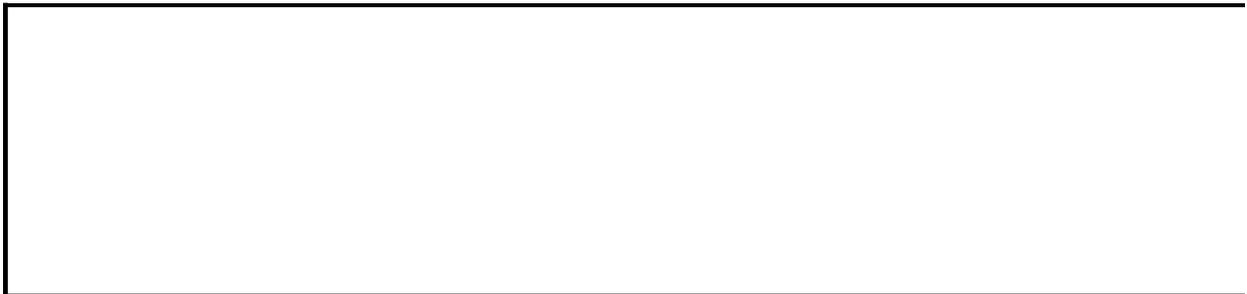
Example

This example adds borders between rows and between columns in the first table of the active document, and then it sets the colors for both the inside and outside borders.

```
If ActiveDocument.Tables.Count >= 1 Then
    Set myTable = ActiveDocument.Tables(1)
    With myTable.Borders
        .InsideLineStyle = True
        .InsideColorIndex = wdBrightGreen
        .OutsideColorIndex = wdPink
    End With
End If
```

This example adds a red, 0.75-point double border around the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).Borders
    .OutsideLineStyle = wdLineStyleDouble
    .OutsideLineWidth = wdLineWidth075pt
    .OutsideColorIndex = wdRed
End With
```



↳ [Show All](#)

OutsideLineStyle Property

Returns or sets the outside border for the specified object. Returns **wdUndefined** if more than one kind of border is applied to the specified object; otherwise, returns **False** or a **WdLineStyle** constant. Can be set to **True**, **False**, or a [WdLineStyle](#) constant.

WdLineStyle can be one of these WdLineStyle constants.

wdLineStyleDashDot

wdLineStyleDashDotDot

wdLineStyleDashDotStroked

wdLineStyleDashLargeGap

wdLineStyleDashSmallGap

wdLineStyleDot

wdLineStyleDouble

wdLineStyleDoubleWavy

wdLineStyleEmboss3D

wdLineStyleEngrave3D

wdLineStyleInset

wdLineStyleNone

wdLineStyleOutset

wdLineStyleSingle

wdLineStyleSingleWavy

wdLineStyleThickThinLargeGap

wdLineStyleThickThinMedGap

wdLineStyleThickThinSmallGap

wdLineStyleThinThickLargeGap

wdLineStyleThinThickMedGap

wdLineStyleThinThickSmallGap

wdLineStyleThinThickThinLargeGap

wdLineStyleThinThickThinMedGap
wdLineStyleThinThickThinSmallGap
wdLineStyleTriple

expression.**OutsideLineStyle**

expression Required. An expression that returns a [Borders](#) object.

Remarks

True sets the line style to the default line style and the line width to the default line width. The default line style and width can be set using the [DefaultBorderLineWidth](#) and [DefaultBorderLineStyle](#) properties.

Use either of the following instructions to remove the outside border from the first table in the active document.

```
ActiveDocument.Tables(1).Borders.OutsideLineStyle = wdLineStyleNone  
ActiveDocument.Tables(1).Borders.OutsideLineStyle = False
```

Example

This example adds a double 0.75-point border around the first paragraph in the active document.

```
With ActiveDocument.Paragraphs(1).Borders
    .OutsideLineStyle = wdLineStyleDouble
    .OutsideLineWidth = wdLineWidth075pt
End With
```

This example adds a border around the first table in the active document.

```
If ActiveDocument.Tables.Count >= 1 Then
    Set myTable = ActiveDocument.Tables(1)
    myTable.Borders.OutsideLineStyle = wdLineStyleSingle
End If
```



▾ [Show All](#)

OutsideLineWidth Property

Returns or sets the line width of the outside border of an object. Returns **wdUndefined** if the object has outside borders with more than one line width; otherwise, returns **False** or a **WdLineWidth** constant. Can be set to **True**, **False**, or a [WdLineWidth](#) constant. Read/write.

WdLineWidth can be one of these WdLineWidth constants.

wdLineWidth025pt

wdLineWidth050pt

wdLineWidth075pt

wdLineWidth100pt

wdLineWidth150pt

wdLineWidth225pt

wdLineWidth300pt

wdLineWidth450pt

wdLineWidth600pt

expression.**OutsideLineWidth**

expression Required. An expression that returns a [Borders](#) object.

Example

This example adds a wavy border around the first table in the active document.

```
If ActiveDocument.Tables.Count >= 1 Then
    With ActiveDocument.Tables(1).Borders
        .OutsideLineStyle = wdLineStyleSingleWavy
        .OutsideLineWidth = wdLineWidth075pt
    End With
End If
```

This example adds dotted borders around the first four paragraphs in the active document.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range(Start:=myDoc.Paragraphs(1).Range.Start, _
    End:=myDoc.Paragraphs(4).Range.End)
myRange.Borders.OutsideLineStyle = wdLineStyleDot
myRange.Borders.OutsideLineWidth = wdLineWidth075pt
```



Overflowing Property

True if the text inside the specified text frame doesn't all fit within the frame.
Read-only **Boolean**.

expression.**Overflowing**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example checks to see whether the text in MyTextBox is overflowing its text frame. If so, the example adds another text box and links the two text boxes so that the text flows into the next one.

```
Set myTBox = ActiveDocument.Shapes("MyTextBox")
If myTBox.TextFrame.Overflowing = True Then
    Set nextTBox = ActiveDocument.Shapes. _
        AddTextbox(msoTextOrientationHorizontal, 72, 72, 100, 200)
    MyTBox.TextFrame.Next = nextTBox.TextFrame
End If
```



↳ [Show All](#)

OverPrint Property

-

When creating separation plates for commercial printing, **MsoTrue** indicates that the specified shape is not printed on the separation plates where the ink level of the shape is set to 0 (zero). Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Does not apply to this property.

msoFalse Removes any color left for the selected shape by earlier plates.

msoTriStateMixed Does not apply to this property.

msoTriStateToggle Does not apply to this property.

msoTrue Excludes the shape from being processed or printed on a CMYK plate.

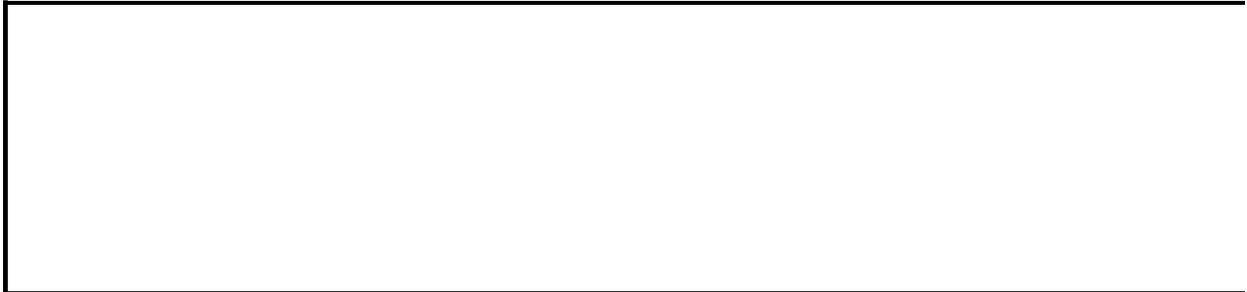
expression.**OverPrint**

expression Required. An expression that returns a [ColorFormat](#) object.

Example

This example creates a new shape in the active document, sets the fill color, and excludes the shape from the printer's plate.

```
Sub TintShade()  
  Dim shpHeart As Shape  
  Set shpHeart = ActiveDocument.Shapes.AddShape( _  
    Type:=msoShapeHeart, Left:=150, _  
    Top:=150, Width:=250, Height:=250)  
  With shpHeart.Fill.ForeColor  
    .SetCMYK Cyan:=0, Magenta:=125, Yellow:=12, Black:=25  
    .TintAndShade = 0.3  
    .OverPrint = msoTrue  
  End With  
End Sub
```



Overtyping Property

True if Overtyping mode is active. In Overtyping mode, the characters you type replace existing characters one by one. When Overtyping isn't active, the characters you type move existing text to the right. Read/write **Boolean**.

expression.**Overtyping**

expression Required. An expression that returns an [Options](#) object.

Example

If Overtyping mode is active, this example displays a message box asking whether Overtyping should be deactivated. If the user clicks the Yes button, Overtyping mode is made inactive.

```
If Options.Overtyping = True Then
    aButton = MsgBox("Overtyping is on. Turn off?", 4)
    If aButton = vbYes Then Options.Overtyping = False
End If
```



OwnHelp Property

-

Specifies the source of the text that's displayed in a message box when a form field has the focus and the user presses F1. If **True**, the text specified by the [HelpText](#) property is displayed. If **False**, the text in the AutoText entry specified by the **HelpText** property is displayed. Read/write **Boolean**.

expression.**OwnHelp**

expression Required. An expression that returns a [FormField](#) object.

Example

This example sets the help text for the first form field in the current section to the contents of the AutoText entry named "Sample."

```
With Selection.Sections(1).Range.FormFields(1)  
    .OwnHelp = False  
    .HelpText = "Sample"  
End With
```



OwnStatus Property

-

Specifies the source of the text that's displayed in the status bar when a form field has the focus. If **True**, the text specified by the [StatusText](#) property is displayed. If **False**, the text of the AutoText entry specified by the **StatusText** property is displayed. Read/write **Boolean**.

expression.**OwnStatus**

expression Required. An expression that returns a [FormField](#) object.

Example

This example sets the status bar text for the form field named "Account" to the contents of the AutoText entry named "Acct."

```
With ActiveDocument.FormFields("Account")  
    .OwnStatus = False  
    .StatusText = "Acct"  
End With
```



PageBreakBefore Property

-
True if a page break is forced before the specified paragraphs. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

Example

This example forces a page break before the first paragraph in the selection.

```
Selection.Paragraphs(1).PageBreakBefore = True
```



PageColumns Property

Returns or sets the number of pages to be displayed side by side on-screen at the same time in print layout view or print preview. Read/write **Long**.

expression.**PageColumns**

expression Required. An expression that returns a [Zoom](#) object.

Example

This example switches the active window to print layout view and displays two pages side by side.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .Zoom.PageColumns = 2
    .Zoom.PageRows = 1
End With
```

This example switches the document window for Hello.doc to print layout view and displays one full page.

```
With Windows("Hello.doc").View
    .Type = wdPrintView
    With .Zoom
        .PageColumns = 1
        .PageRows = 1
        .PageFit = wdPageFitFullPage
    End With
End With
```



PageDesign Property

Returns or sets the name of the template attached to the document created by the Letter Wizard. Read/write **String**.

expression.**PageDesign**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, includes the header and footer from the Contemporary Letter template, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
With myContent
    .PageDesign = "C:\MSOffice\Templates\" _
        & "Letters & Faxes\Contemporary Letter.dot"
    .IncludeHeaderFooter = True
End With
Documents.Add.RunLetterWizard LetterContent:=myContent
```



↳ [Show All](#)

PageFit Property

Returns or sets the view magnification of a window so that either the entire page is visible or the entire width of the page is visible. Read/write [WdPageFit](#).

WdPageFit can be one of these WdPageFit constants.

wdPageFitBestFit

wdPageFitFullPage

wdPageFitNone

wdPageFitTextFit

expression.**PageFit**

expression Required. An expression that returns a [Zoom](#) object.

Remarks

The **wdPageFitFullPage** constant has no effect if the document isn't in print view.

When the **PageFit** property is set to **wdPageFitBestFit**, the zoom percentage is automatically recalculated every time the document window size is changed. Setting this property to **wdPageFitNone** keeps the zoom percentage from being recalculated whenever this happens.

Example

This example changes the magnification percentage of the window for Letter.doc so that the entire width of the text is visible.

```
With Windows("Letter.doc").View
    .Type = wdNormalView
    .Zoom.PageFit = wdPageFitBestFit
End With
```

This example switches the active window to print view and changes the magnification so that the entire page is visible.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .Zoom.PageFit = wdPageFitFullPage
End With
```



PageHeight Property

Returns or sets the height of the page in points. Read/write **Single**.

expression.**PageHeight**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

Setting the **PageHeight** property changes the [PaperSize](#) property to **wdPaperCustom**.

Use the **PaperSize** property to set the page height and width to those of a predefined paper size, such as Letter or A4.

Example

This example sets the page height for the active document to 9 inches.

```
With ActiveDocument.PageSetup  
    .PageHeight = InchesToPoints(9)  
    .PageWidth = InchesToPoints(7)  
End With
```



PageNumbers Property

Returns a [PageNumbers](#) collection that represents all the page number fields included in the specified header or footer.

expression.**PageNumbers**

expression Required. An expression that returns a [HeaderFooter](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a new document and adds page numbers to the footer.

```
Set myDoc = Documents.Add
With myDoc.Sections(1).Footers(wdHeaderFooterPrimary)
    .PageNumbers.Add PageNumberAlignment := wdAlignPageNumberCenter
End With
```



PageNumberSeparator Property

Returns or sets the characters (up to five) that separate individual page references in a table of authorities. The default is a comma and a space. Corresponds to the \l switch for a Table of Authorities (TOA) field. Read/write **String**.

expression.**PageNumberSeparator**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example formats the tables of authorities in the active document to use a comma as the page separator (for example, "9,12").

```
For Each myTOA In ActiveDocument.TablesOfAuthorities
    myTOA.PageNumberSeparator = ","
Next myTOA
```



PageRangeSeparator Property

Returns or sets the characters (up to five) that separate a range of pages in a table of authorities. The default is an en dash. Corresponds to the \g switch for a Table of Authorities (TOA) field. Read/write **String**.

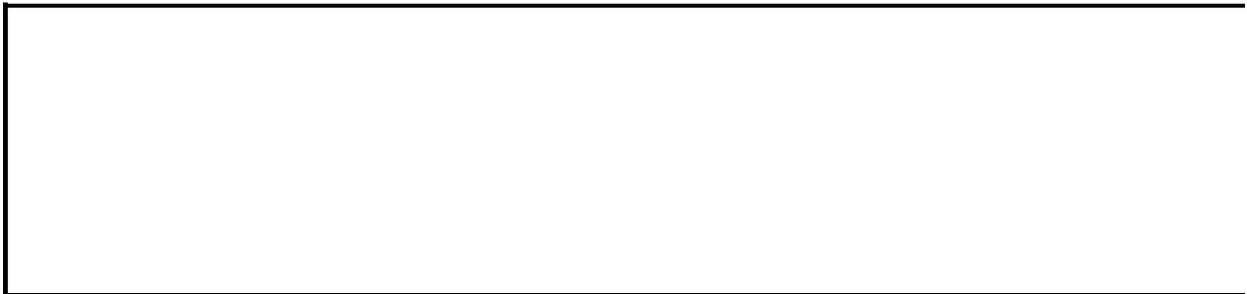
expression.**PageRangeSeparator**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example formats the first table of authorities in the active document to use a hyphen with a space on either side as the page range separator (for example, "9 - 12").

```
ActiveDocument.TablesOfAuthorities(1).PageRangeSeparator = " - "
```



PageRows Property

Returns or sets the number of pages to be displayed one above the other on-screen at the same time in print layout view or print preview. Read/write **Long**.

expression.**PageRows**

expression Required. An expression that returns a [Zoom](#) object.

Example

This example switches the active window to print preview and displays two pages one above the other.

```
PrintPreview = True  
With ActiveDocument.ActiveWindow.View.Zoom  
    .PageColumns = 1  
    .PageRows = 2  
End With
```



PageSetup Property

-

Returns a [PageSetup](#) object that's associated with the specified document, range, section, sections, or selection. Read-only.

Example

This example sets the right margin of the active document to 72 points (1 inch).

```
ActiveDocument.PageSetup.RightMargin = InchesToPoints(1)
```

This example sets the gutter for the first section in Summary.doc to 36 points (0.5 inch).

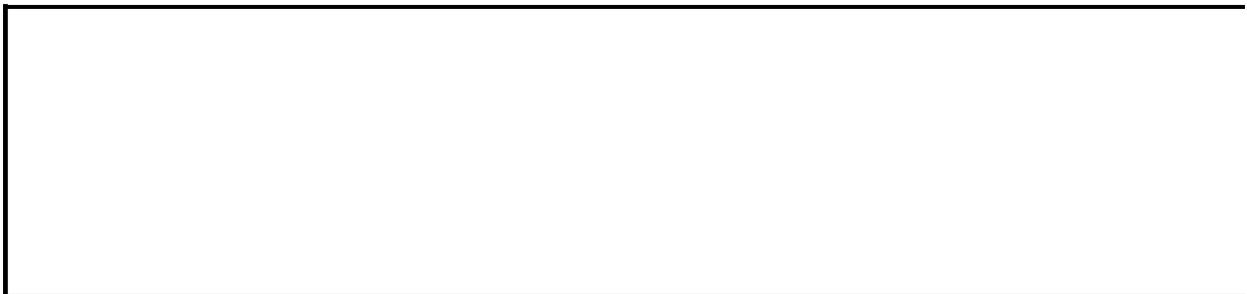
```
Documents("Summary.doc").Sections(1).PageSetup.Gutter = 36
```

This example sets the header and footer distance to 18 points (0.25 inch) from the top and bottom of the page, respectively. This formatting change is applied to the section that contains the selection.

```
With Selection.PageSetup  
    .FooterDistance = 18  
    .HeaderDistance = 18  
End With
```

This example displays the left margin setting, in inches.

```
MsgBox PointsToInches(ActiveDocument.PageSetup.LeftMargin) _  
    & " inches"
```



↳ [Show All](#)

PageSize Property

Returns or sets the page size for the specified custom mailing label. Read/write [WdCustomLabelPageSize](#).

WdCustomLabelPageSize can be one of these WdCustomLabelPageSize constants.

wdCustomLabelA4

wdCustomLabelA4LS

wdCustomLabelA5

wdCustomLabelA5LS

wdCustomLabelB4JIS

wdCustomLabelB5

wdCustomLabelFanfold

wdCustomLabelHigaki

wdCustomLabelHigakiLS

wdCustomLabelLetter

wdCustomLabelLetterLS

wdCustomLabelMini

wdCustomLabelVertHalfSheet

wdCustomLabelVertHalfSheetLS

expression.**PageSize**

expression Required. An expression that returns a [CustomLabel](#) object.

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 creates a new custom label named "Home Address" and then sets various properties for the label, including the page size.

```
Set myLabel = Application.MailingLabel _  
    .CustomLabels.Add(Name:="Home Address", DotMatrix:=False)  
With myLabel  
    .Height = InchesToPoints(0.5)  
    .HorizontalPitch = InchesToPoints(2.06)  
    .NumberAcross = 4  
    .NumberDown = 20  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0.28)  
    .TopMargin = InchesToPoints(0.5)  
    .VerticalPitch = InchesToPoints(0.5)  
    .Width = InchesToPoints(1.75)  
End With
```



PageWidth Property

Returns or sets the width of the page in points. Read/write **Single**.

expression.**PageWidth**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

Setting the **PageWidth** property changes the [PaperSize](#) property to **wdPaperCustom**.

Use the **PaperSize** property to set the page height and width to those of a predefined paper size, such as Letter or A4.

Example

This example returns the page width for Document1. The [PointsToInches](#) method is used to convert points to inches.

```
Set doc1set = Documents("Document1").PageSetup
Msgbox "The page width is " _
    & PointsToInches(doc1set.PageWidth) & " inches."
```



Pagination Property

True if Microsoft Word repaginates documents in the background. Read/write **Boolean**.

expression.**Pagination**

expression Required. An expression that returns a [Options](#) object.

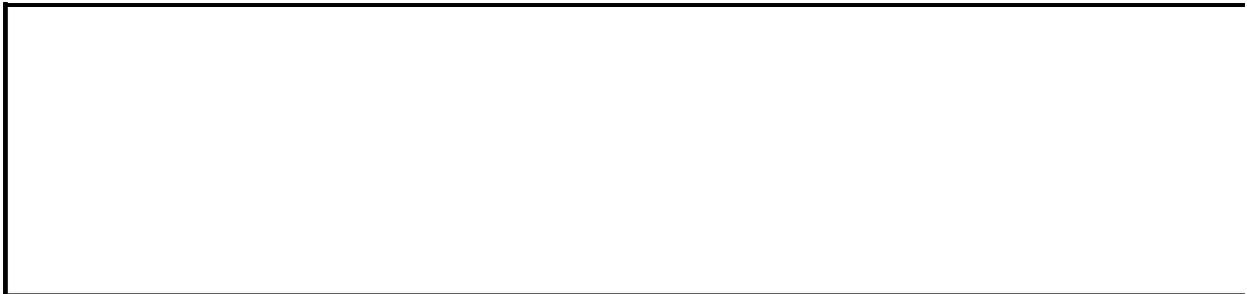
Example

This example sets Word to perform background repagination.

```
Options.Pagination = True
```

This example returns the current status of the **Background repagination** option on the **General** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.Pagination
```



Panes Property

Returns a [Panes](#) collection that represents all the window panes for the specified window.

expression.**Panes**

expression Required. An expression that returns a [Window](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example splits the active window in half.

```
If ActiveDocument.ActiveWindow.Panes.Count = 1 Then _  
    ActiveDocument.ActiveWindow.Panes.Add
```

This example activates the first pane in the window for Document2.

```
Windows("Document2").Panes(1).Activate
```



↳ [Show All](#)

PaperSize Property

Returns or sets the paper size. Read/write [WdPaperSize](#).

WdPaperSize can be one of these WdPaperSize constants.

wdPaper10x14

wdPaper11x17

wdPaperA3

wdPaperA4

wdPaperA4Small

wdPaperA5

wdPaperB4

wdPaperB5

wdPaperCSheet

wdPaperCustom

wdPaperDSheet

wdPaperEnvelope10

wdPaperEnvelope11

wdPaperEnvelope12

wdPaperEnvelope14

wdPaperEnvelope9

wdPaperEnvelopeB4

wdPaperEnvelopeB5

wdPaperEnvelopeB6

wdPaperEnvelopeC3

wdPaperEnvelopeC4

wdPaperEnvelopeC5

wdPaperEnvelopeC6

wdPaperEnvelopeC65

wdPaperEnvelopeDL

wdPaperEnvelopeItaly
wdPaperEnvelopeMonarch
wdPaperEnvelopePersonal
wdPaperESheet
wdPaperExecutive
wdPaperFanfoldLegalGerman
wdPaperFanfoldStdGerman
wdPaperFanfoldUS
wdPaperFolio
wdPaperLedger
wdPaperLegal
wdPaperLetter
wdPaperLetterSmall
wdPaperNote
wdPaperQuarto
wdPaperStatement
wdPaperTabloid

expression.**PaperSize**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

Setting the [PageHeight](#) or [PageWidth](#) property changes the **PaperSize** property to **wdPaperCustom**.

Example

This example sets the paper size to legal for the first document.

```
Documents(1).PageSetup.PaperSize = wdPaperLegal
```



ParagraphFormat Property

-

Returns or sets a [ParagraphFormat](#) object that represents the paragraph settings for the specified range, selection, find or replacement operation, or style.
Read/write.

Example

This example sets the paragraph formatting for the current selection to be right-aligned.

```
Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
```

This example sets paragraph formatting for a range that includes the entire contents of MyDoc.doc. Paragraphs in this document are double-spaced and have a custom tab stop at 0.25 inch.

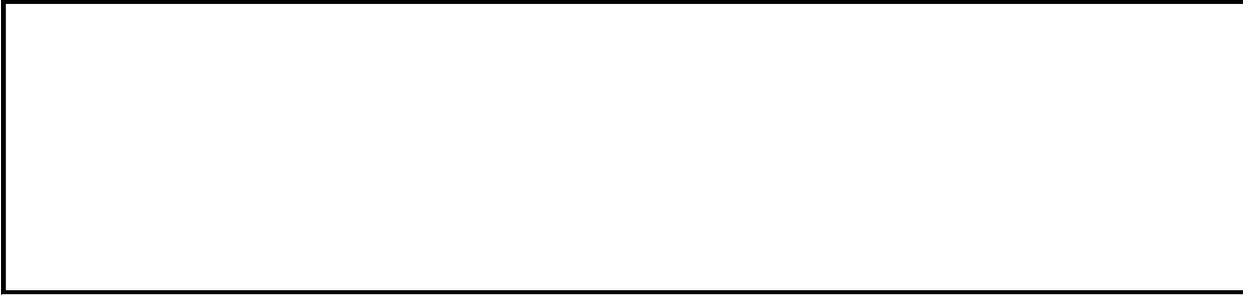
```
Set myRange = Documents("MyDoc.doc").Content  
With myRange.ParagraphFormat  
    .Space2  
    .TabStops.Add Position:=InchesToPoints(.25)  
End With
```

This example modifies the Heading 2 style for the active document. Paragraphs formatted with this style are indented to the first tab stop and double-spaced.

```
With ActiveDocument.Styles(wdStyleHeading2).ParagraphFormat  
    .TabIndent(1)  
    .Space2  
End With
```

This example finds all double-spaced paragraphs in the active document and replaces the formatting with 1.5-line spacing.

```
With ActiveDocument.Content.Find  
    .ClearFormatting  
    .ParagraphFormat.Space2  
    .Replacement.ClearFormatting  
    .Replacement.ParagraphFormat.Space15  
    .Execute FindText:="", ReplaceWith:"", _  
        Replace:=wdReplaceAll  
End With
```



Paragraphs Property

Returns a [Paragraphs](#) collection that represents all the paragraphs in the specified document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the line spacing to single for the collection of all paragraphs in section one in the active document.

```
ActiveDocument.Sections(1).Range.Paragraphs.LineSpacingRule = _  
    wdLineSpaceSingle
```

This example sets the line spacing to double for the first paragraph in the selection.

```
Selection.Paragraphs(1).LineSpacingRule = wdLineSpaceDouble
```



Parent Property

-

For the [TextFrame](#) object, returns a [Shape](#) object representing the parent shape of the text frame. For all other objects, returns an object that represents the parent object of the specified object.

expression.**Parent**

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

Example

This example sets a variable to the parent object of the **Bookmarks** object and displays a message box with the object type name of the variable.

```
Set myObject = ActiveDocument.Bookmarks.Parent  
MsgBox TypeName(myObject)
```

This example sets a variable to the first cell in the first table of the active document, changes the width of the cell to 36 points, and removes borders from the table.

```
Set myRange = ActiveDocument.Tables(1).Cell(1, 1)  
With myRange  
    .SetWidth ColumnWidth:=36, RulerStyle:=wdAdjustNone  
    .Parent.Borders.Enable = False  
End With
```



ParentFrameset Property

Returns a [Frameset](#) object that represents the parent of the specified **Frameset** object on a frames page.

expression.**ParentFrameset**

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

Remarks

For more information on creating frames pages, see [Creating frames pages](#).

Example

This example returns the number of child **Frameset** objects belonging to the parent **Frameset** object of the specified frame.

```
MsgBox ActiveDocument.ActiveWindow.ActivePane _  
    .Frameset.ParentFrameset.ChildFramesetCount
```



ParentGroup Property

Returns a [Shape](#) object that represents the common parent shape of a child shape or a range of child shapes.

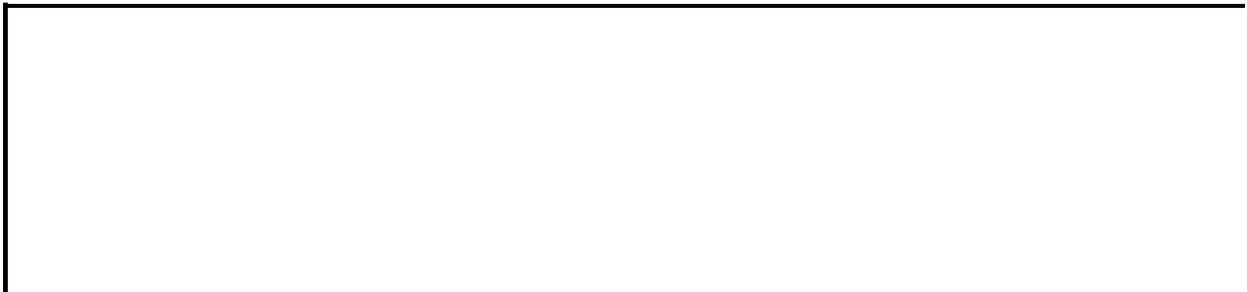
expression.**ParentGroup**

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 color. This example assumes that the active document does not currently contain any shapes. If it does, an error may occur.

```
Sub ParentGroupShape()  
    Dim pgShape As Shape  
  
    'Add two shapes to active document and group  
    With ActiveDocument.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 pgShape = ActiveDocument.Shapes(1) _  
        .GroupItems(1).ParentGroup  
    pgShape.Fill.ForeColor.RGB = RGB(Red:=100, Green:=0, Blue:=255)  
  
End Sub
```



PartOfSpeechList Property

Returns a list of the parts of speech corresponding to the meanings found for the word or phrase looked up in the thesaurus. The list is returned as an array of integers. Read-only **Variant**.

expression.**PartOfSpeechList**

expression Required. An expression that returns a [SynonymInfo](#) object.

Remarks

The list of the parts of speech is returned as an array consisting of the following **WdPartOfSpeech** constants: **wdAdjective**, **wdAdverb**, **wdConjunction**, **wdIdiom**, **wdInterjection**, **wdNoun**, **wdOther**, **wdPreposition**, **wdPronoun**, and **wdVerb**. The array elements are ordered to correspond to the elements returned by the [MeaningList](#) property.

Example

This example checks to see whether the thesaurus found any meanings for the selection. If so, the meanings and their corresponding parts of speech are displayed in a series of message boxes.

```
Set mySynInfo = Selection.Range.SynonymInfo
If mySynInfo.MeaningCount <> 0 Then
    myList = mySynInfo.MeaningList
    myPos = mySynInfo.PartOfSpeechList
    For i = 1 To UBound(myPos)
        Select Case myPos(i)
            Case wdAdjective
                pos = "adjective"
            Case wdNoun
                pos = "noun"
            Case wdAdverb
                pos = "adverb"
            Case wdVerb
                pos = "verb"
            Case Else
                pos = "other"
        End Select
        MsgBox myList(i) & " found as " & pos
    Next i
Else
    MsgBox "There were no meanings found."
End If
```



Passim Property

-
True if five or more page references to the same authority are replaced with "Passim." Corresponds to the \p switch for a Table of Authorities (TOA) field.
Read/write **Boolean**.

expression.**Passim**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

This example formats the first table of authorities in Brief.doc to use page references instead of "Passim."

```
Documents("Brief.doc").TablesOfAuthorities(1).Passim = False
```

This example formats the tables of authorities in the active document to replace each instance of five or more page references for the same entry with "Passim."

```
For Each myTOA In ActiveDocument.TablesOfAuthorities  
    myToa.Passim = True  
Next myTOA
```



Password Property

-

Sets a password that must be supplied to open the specified document. Write-only **String**.

Example

This example opens Earnings.doc, sets a password for it, and then closes the document.

```
Set myDoc = Documents _  
    .Open(FileName:="C:\My Documents\Earnings.doc")  
myDoc.Password = "why"  
myDoc.Close
```



PasswordEncryptionAlgorithm Property

Returns a **String** indicating the algorithm Microsoft Word uses for encrypting documents with passwords. Read-only.

expression.**PasswordEncryptionAlgorithm**

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

Remarks

Use the [SetPasswordEncryptionOptions](#) method to specify the algorithm Word uses for encrypting documents with passwords.

Example

This example sets the password encryption options if the password encryption algorithm in use is "OfficeXor," which is the password algorithm used in versions of Word prior to Word 97 for Windows.

```
Sub PasswordSettings()  
  With ActiveDocument  
    If .PasswordEncryptionAlgorithm = "OfficeXor" Then  
      .SetPasswordEncryptionOptions _  
        PasswordEncryptionProvider:="Microsoft RSA SChannel  
        PasswordEncryptionAlgorithm:="RC4", _  
        PasswordEncryptionKeyLength:=56, _  
        PasswordEncryptionFileProperties:=True  
    End If  
  End With  
End Sub
```



PasswordEncryptionFileProperties Property

True if Microsoft Word encrypts file properties for password-protected documents. Read-only **Boolean**.

expression.**PasswordEncryptionFileProperties**

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

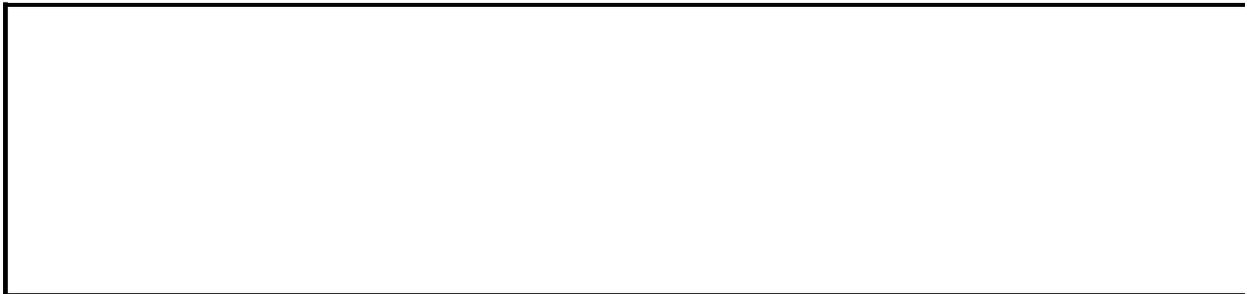
Remarks

Use the [SetPasswordEncryptionOptions](#) method to specify whether Word encrypts file properties for password-protected documents.

Example

This example sets the password encryption options if the file properties are not encrypted for password-protected documents.

```
Sub PasswordSettings()  
  With ActiveDocument  
    If .PasswordEncryptionFileProperties = False Then  
      .SetPasswordEncryptionOptions _  
        PasswordEncryptionProvider:="Microsoft RSA SChannel  
        PasswordEncryptionAlgorithm:="RC4", _  
        PasswordEncryptionKeyLength:=56, _  
        PasswordEncryptionFileProperties:=True  
    End If  
  End With  
End Sub
```



PasswordEncryptionKeyLength Property

Returns a **Long** indicating the key length of the algorithm Microsoft Word uses when encrypting documents with passwords. Read-only.

expression.**PasswordEncryptionKeyLength**

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

Remarks

Use the [SetPasswordEncryptionOptions](#) method to specify the key length Word uses when encrypting documents with passwords.

Example

This example sets the password encryption options if the password encryption key length is less than 40.

```
Sub PasswordSettings()  
  With ActiveDocument  
    If .PasswordEncryptionKeyLength < 40 Then  
      .SetPasswordEncryptionOptions _  
        PasswordEncryptionProvider:="Microsoft RSA SChannel  
        PasswordEncryptionAlgorithm:="RC4", _  
        PasswordEncryptionKeyLength:=56, _  
        PasswordEncryptionFileProperties:=True  
    End If  
  End With  
End Sub
```



PasswordEncryptionProvider Property

Returns a **String** specifying the name of the algorithm encryption provider that Microsoft Word uses when encrypting documents with passwords. Read-only.

expression.**PasswordEncryptionProvider**

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

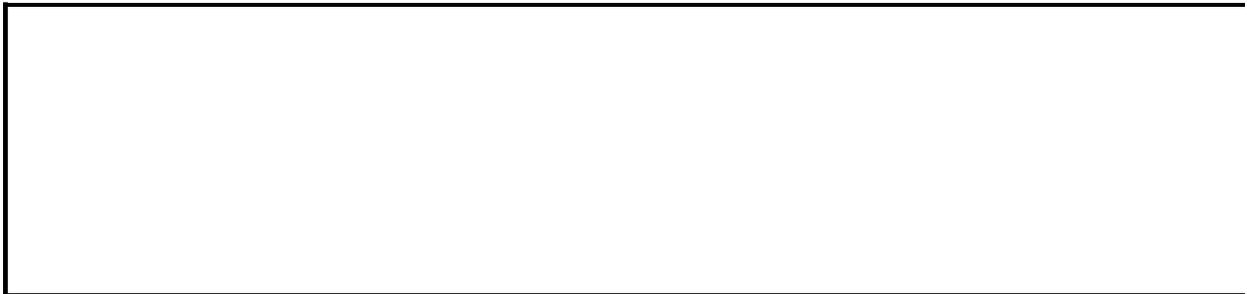
Remarks

Use the [SetPasswordEncryptionOptions](#) method to specify the name of the algorithm encryption provider Word uses when encrypting documents with passwords.

Example

This example sets the password encryption options if the password encryption algorithm in use is not "Microsoft RSA SChannel Cryptographic Provider."

```
Sub PasswordSettings()  
  With ActiveDocument  
    If .PasswordEncryptionProvider <> "Microsoft RSA SChannel Cr  
      .SetPasswordEncryptionOptions _  
        PasswordEncryptionProvider:="Microsoft RSA SChannel  
        PasswordEncryptionAlgorithm:="RC4", _  
        PasswordEncryptionKeyLength:=56, _  
        PasswordEncryptionFileProperties:=True  
    End If  
  End With  
End Sub
```



PasteAdjustParagraphSpacing Property

True if Microsoft Word automatically adjusts the spacing of paragraphs when cutting and pasting selections. Read/write **Boolean**.

expression.**PasteAdjustParagraphSpacing**

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

Example

This example sets Word to automatically adjust the spacing of paragraphs when cutting and pasting selections if the option has been disabled.

```
Sub AdjustParaSpace()  
  With Options  
    If .PasteAdjustParagraphSpacing = False Then  
      .PasteAdjustParagraphSpacing = True  
    End If  
  End With  
End Sub
```



PasteAdjustTableFormatting Property

True if Microsoft Word automatically adjusts the formatting of tables when cutting and pasting selections. Read/write **Boolean**.

expression.**PasteAdjustTableFormatting**

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

Example

This example sets Word to automatically adjust the formatting of tables when cutting and pasting if the option has been disabled.

```
Sub AdjustTableFormatting()  
  With Options  
    If .PasteAdjustTableFormatting = False Then  
      .PasteAdjustTableFormatting = True  
    End If  
  End With  
End Sub
```



PasteAdjustWordSpacing Property

True if Microsoft Word automatically adjusts the spacing of words when cutting and pasting selections. Read/write **Boolean**.

expression.**PasteAdjustWordSpacing**

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

Example

This example sets Word to automatically adjust the spacing of words when cutting and pasting selections if the option has been disabled.

```
Sub AdjustWordSpace()  
  With Options  
    If .PasteAdjustWordSpacing = False Then  
      .PasteAdjustWordSpacing = True  
    End If  
  End With  
End Sub
```



PasteMergeFromPPT Property

True to merge text formatting when pasting from Microsoft PowerPoint.
Read/write **Boolean**.

expression.**PasteMergeFromPPT**

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

Example

This example sets Microsoft Word to automatically merge text formatting when pasting content from PowerPoint if the option has been disabled.

```
Sub AdjustPPTFormatting()  
  With Options  
    If .PasteMergeFromPPT = False Then  
      .PasteMergeFromPPT = True  
    End If  
  End With  
End Sub
```



PasteMergeFromXL Property

True to merge table formatting when pasting from Microsoft Excel. Read/write **Boolean**.

expression.**PasteMergeFromXL**

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

Example

This example sets Microsoft Word to automatically merge table formatting when pasting Excel tables if the option has been disabled.

```
Sub AdjustExcelFormatting()  
  With Options  
    If .PasteMergeFromXL = False Then  
      .PasteMergeFromXL = True  
    End If  
  End With  
End Sub
```



PasteMergeLists Property

True to merge the formatting of pasted lists with surrounding lists. Read/write **Boolean**.

expression.**PasteMergeLists**

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

Example

This example sets Microsoft Word to automatically merge list formatting with surrounding lists if the option has been disabled.

```
Sub UseSmartStyle()  
  With Options  
    If .PasteMergeLists = False Then  
      .PasteMergeLists = True  
    End If  
  End With  
End Sub
```



PasteSmartCutPaste Property

True if Microsoft Word intelligently pastes selections into a document.
Read/write **Boolean**.

expression.**PasteSmartCutPaste**

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

Example

This example sets Word to enable intelligent selection pasting if the option has been disabled.

```
Sub EnableSmartCutPaste()  
    If Options.PasteSmartCutPaste = False Then  
        Options.PasteSmartCutPaste = True  
    End If  
End Sub
```



PasteSmartStyleBehavior Property

True if Microsoft Word intelligently merges styles when pasting a selection from a different document. Read/write **Boolean**.

expression.**PasteSmartStyleBehavior**

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

Example

This example sets Word to intelligently paste styles in text selected from a different document if the option has been disabled.

```
Sub UseSmartStyle()  
    With Options  
        If .PasteSmartStyleBehavior = False Then  
            .PasteSmartStyleBehavior = True  
        End If  
    End With  
End Sub
```



Path Property

Returns the disk or Web path to the specified object. Read-only **String**.

expression.**Path**

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

Remarks

The path doesn't include a trailing character — for example, "C:\MSOffice" or "http://MyServer". Use the [PathSeparator](#) property to add the character that separates folders and drive letters. Use the [Name](#) property to return the file name without the path and use the [FullName](#) property to return the file name and the path together.

Note You can use the **PathSeparator** property to build Web addresses even though they contain forward slashes (/) and the **PathSeparator** property defaults to a backslash (\).

Example

This example displays the path and file name of the active document.

```
MsgBox ActiveDocument.Path & Application.PathSeparator & _  
ActiveDocument.Name
```

This example changes the current folder to the path of the template attached to the active document.

```
ChDir ActiveDocument.AttachedTemplate.Path
```

This example displays the path of the first add-in in the **AddIns** collection.

```
If AddIns.Count >= 1 Then MsgBox AddIns(1).Path
```



PathSeparator Property

Returns the character used to separate folder names. This property returns a backslash (\). Read-only **String**.

expression.**PathSeparator**

expression Required. An expression that returns an [Application](#) object.

Remarks

You can use **PathSeparator** property to build Web addresses even though they contain forward slashes (/).

The **FullName** property returns the path and file name as a single string.

Example

This example displays the path and file name of the active document.

```
MsgBox ActiveDocument.Path & Application.PathSeparator & _  
    ActiveDocument.Name
```

If the first add-in is a template, this example unloads the template and opens it.

```
If Addins(1).Compiled = False Then  
    Addins(1).Installed = False  
    Documents.Open FileName:=AddIns(1).Path _  
        & Application.PathSeparator _  
        & AddIns(1).Name  
End If
```



↳ [Show All](#)

Pattern Property

Returns or sets a value that represents the pattern applied to the specified fill or line. Read-only [MsoPatternType](#) for the [FillFormat](#) object; read/write [MsoPatternType](#) 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 objects in the Applies To list.

Remarks

You can also use the [Patterned](#) method to set the pattern for the fill or line.

Use the [BackColor](#) and [ForeColor](#) properties to set the colors used in the pattern.

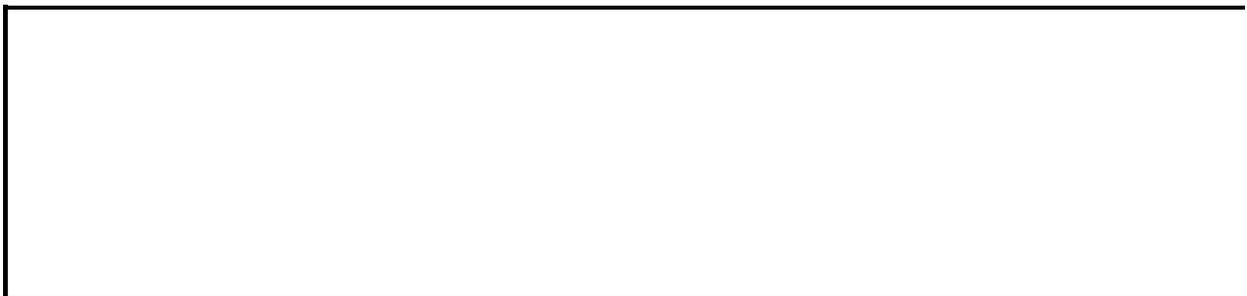
Example

This example adds a rectangle to myDocument and sets its fill pattern to match that of the shape named "rect1." The new rectangle has the same pattern as "rect1" but not necessarily the same colors. The colors used in the pattern are set with the **BackColor** and **ForeColor** properties.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    pattern1 = .Item("rect1").Fill.Pattern
    With .AddShape(msoShapeRectangle, 100, 100, 120, 80).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .BackColor.RGB = RGB(0, 0, 255)
        .Patterned pattern1
    End With
End With
```

This example adds a patterned line to myDocument.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddLine(10, 100, 250, 0).Line
    .Weight = 6
    .ForeColor.RGB = RGB(0, 0, 255)
    .BackColor.RGB = RGB(128, 0, 0)
    .Pattern = msoPatternDarkDownwardDiagonal
End With
```



Percentage Property

Returns or sets the magnification for a window as a percentage. Read/write **Long**.

expression.**Percentage**

expression Required. An expression that returns a [Zoom](#) object.

Example

This example switches the active window to normal view and sets the magnification to 80 percent.

```
With ActiveDocument.ActiveWindow.View
    .Type = wdNormalView
    .Zoom.Percentage = 80
End With
```

This example increases the magnification of the active window by 10 percent.

```
Set myZoom = ActiveDocument.ActiveWindow.View.Zoom
myZoom.Percentage = myZoom.Percentage + 10
```



PercentWidth Property

Returns or sets the length of the specified horizontal line expressed as a percentage of the window width. Read/write **Single**.

expression.**PercentWidth**

expression Required. An expression that returns a [HorizontalLineFormat](#) object.

Remarks

Setting this property also sets the [WidthType](#) property to **wdHorizontalLinePercentWidth**.

Example

This example adds a horizontal line and sets its length to 50% of the window width.

```
Selection.InlineShapes.AddHorizontalLineStandard  
ActiveDocument.InlineShapes(1) _  
    .HorizontalLineFormat.PercentWidth = 50
```



↳ [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 a [ThreeDFormat](#) object.

Example

This example sets the extrusion depth for shape one on myDocument to 100 points and specifies that the extrusion be parallel, or orthographic.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .Depth = 100
    .Perspective = msoFalse
End With
```



PictureBullet Property

Returns an [InlineShape](#) object that represents a picture bullet.

expression.**PictureBullet**

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

Example

This example returns the picture bullet for the first list in the active document and sets the picture bullet's width to one-quarter inch. To see this example, first run the code example for the [ApplyPictureBullet](#) method.

```
Sub PicBullet()  
    ActiveDocument.ListTemplates(1) _  
        .ListLevels(1) _  
            .PictureBullet.Width = InchesToPoints(0.25)  
End Sub
```



PictureEditor Property

Returns or sets the name of the application to use to edit pictures. Read/write **String**.

expression.**PictureEditor**

expression Required. An expression that returns an [Options](#) object.

Remarks

You must use the exact wording displayed in the **Picture editor** box on the **Edit** tab of the **Options** dialog box (**Tools** menu). Otherwise, the default setting "Microsoft Word" is used.

If the name of your graphics application doesn't appear in the list, contact the manufacturer of the graphics application for instructions.

Example

This example sets the application used to edit pictures.

```
Options.PictureEditor = "Microsoft Word"
```

This example returns the name of the application to use to edit pictures.

```
MsgBox Options.PictureEditor
```



PictureFormat Property

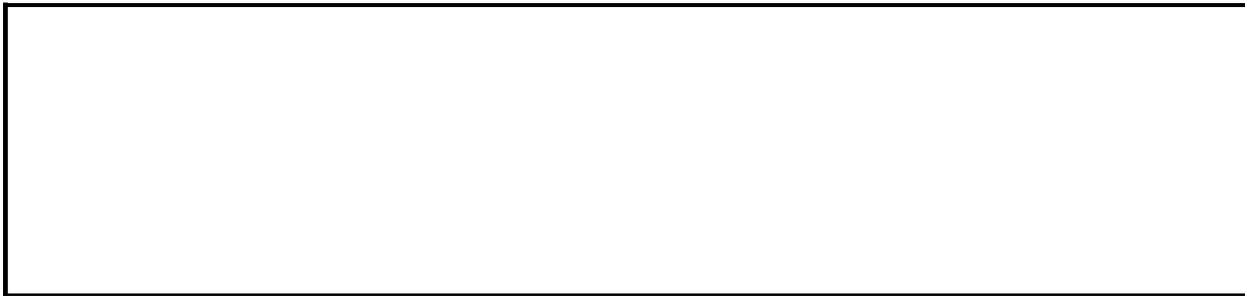
-

Returns a [PictureFormat](#) object that contains picture formatting properties for the specified object. Applies to **Shape**, **ShapeRange**, or **InlineShape** objects that represent pictures or OLE objects. Read-only.

Example

This example sets the brightness and contrast for shape one on myDocument. Shape one must be a picture or an OLE object.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = .75
End With
```



↳ [Show All](#)

PictureWrapType Property

Sets or returns a [WdWrapTypeMerged](#) that indicates how Microsoft Word wraps text around pictures. Read/write.

WdWrapTypeMerged can be one of these WdWrapTypeMerged constants.

wdWrapMergeBehind

wdWrapMergeFront

wdWrapMergeInline Default

wdWrapMergeSquare

wdWrapMergeThrough

wdWrapMergeTight

wdWrapMergeTopBottom

expression.**PictureWrapType**

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

Remarks

This is a default option setting and affects all pictures inserted unless picture wrapping is individually defined for a picture.

Example

This example sets Word to insert and paste all pictures inline with the text if inline is not already specified.

```
Sub PicWrap()  
  With Application.Options  
    If .PictureWrapType <> wdWrapMergeInline Then  
      .PictureWrapType = wdWrapMergeInline  
    End If  
  End With  
End Sub
```



PixelsPerInch Property

Returns or sets the density (pixels per inch) of graphics images and table cells on a Web page. The range of settings is usually from 19 to 480, and common settings for popular screen sizes are 72, 96, and 120. The default setting is 96. Read/write **Long**.

expression.**PixelsPerInch**

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

Remarks

This property determines the size of the images and cells on the specified Web page relative to the size of text whenever you view the saved document in a Web browser. The physical dimensions of the resulting image or cell are the result of the original dimensions (in inches) multiplied by the number of pixels per inch.

Use the [ScreenSize](#) property to set the optimum screen size for the targeted Web browsers.

Example

This example sets the pixel density depending on the target screen size of the Web browser.

```
With Application.DefaultWebOptions
  Select Case .ScreenSize
    Case msoScreenSize800x600
      .PixelsPerInch = 72
    Case msoScreenSize1024x768
      .PixelsPerInch = 96
    Case Else
      .PixelsPerInch = 120
  End Select
End With
```



PlainTextStyle Property

Returns the [Style](#) object that represents the text attributes for e-mail messages that are sent or received using plain text.

expression.**PlainTextStyle**

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

Example

This example sets the plain text font for e-mail messages to Tahoma, size 10.

```
Sub PlainTxt()  
    With Application.EmailOptions.PlainTextStyle  
        .Font.Name = "Tahoma"  
        .Font.Size = 10  
    End With  
End Sub
```



↳ [Show All](#)

Points Property

-

Returns the position of the specified node as a [coordinate pair](#). Each coordinate is expressed in points. Read-only **Variant**.

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 three on myDocument to the right 200 points and down 300 points. Shape three must be a freeform drawing.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3).Nodes
    pointsArray = .Item(2).Points
    currXvalue = pointsArray(1, 1)
    currYvalue = pointsArray(1, 2)
    .SetPosition 2, currXvalue + 200, currYvalue + 300
End With
```



PortraitFontNames Property

Returns a [FontNames](#) object that includes the names of all the available portrait fonts.

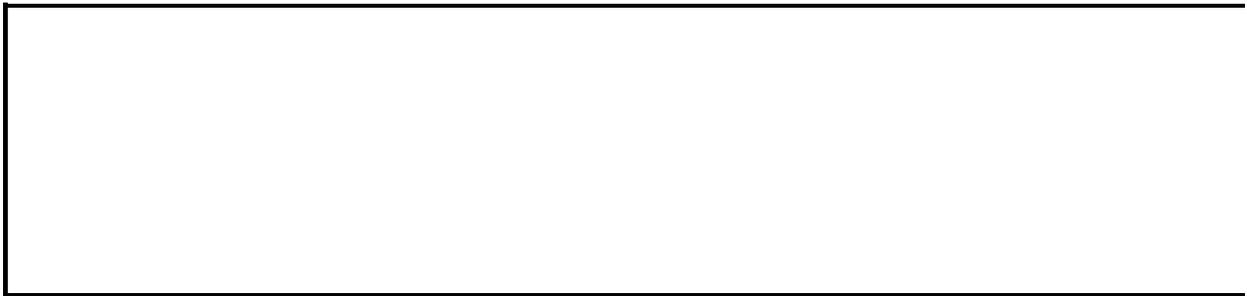
expression.**PortraitFontNames**

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

Example

This example inserts a list of portrait fonts at the insertion point.

```
For Each aFont In PortraitFontNames
  With Selection
    .Collapse Direction:=wdCollapseEnd
    .InsertAfter aFont
    .InsertParagraphAfter
    .Collapse Direction:=wdCollapseEnd
  End With
Next aFont
```



↳ [Show All](#)

Position Property

▶ [Position property as it applies to the **CaptionLabel** object.](#)

Returns or sets the position of caption label text. Read/write [WdCaptionPosition](#).

WdCaptionPosition can be one of these WdCaptionPosition constants.

wdCaptionPositionAbove

wdCaptionPositionBelow

expression.**Position**

expression Required. An expression that returns one of the above objects.

▶ [Position property as it applies to the **DropCap** object.](#)

Returns or sets the position of a dropped capital letter. Read/write [WdDropPosition](#).

WdDropPosition can be one of these WdDropPosition constants.

wdDropNone

wdDropMargin

wdDropNormal

expression.**Position**

expression Required. An expression that returns one of the above objects.

▶ [Position property as it applies to the **TabStop** object.](#)

Returns or sets the position of a tab stop relative to the left margin. Read/write **Single**.

expression.**Position**

expression Required. An expression that returns one of the above objects.

▶ [Position property as it applies to the **Font** object.](#)

Returns or sets the position of text (in points) relative to the base line. A positive number raises the text, and a negative number lowers it. Read/write **Long**.

expression.**Position**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Font** object.](#)

This example lowers the selected text by 2 points.

```
Selection.Font.Position = -2
```

▶ [As it applies to the **TabStop** object.](#)

This example adds a right tab stop to the selected paragraphs 2 inches from the left margin. The position of the tab stop is then displayed in a message box.

```
With Selection.Paragraphs.TabStops
    .ClearAll
    .Add Position:=InchesToPoints(2), Alignment:=wdAlignTabRight
    MsgBox .Item(1).Position & " or " & _
        PointsToInches(.Item(1).Position) & " inches"
End With
```

▶ [As it applies to the **DropCap** object.](#)

This example sets the first paragraph in the active document to begin with a dropped capital letter. The position of the **DropCap** object is set to **wdDropNormal**.

```
With ActiveDocument.Paragraphs(1).DropCap
    .Enable
    .FontName= "Arial"
    .Position = wdDropNormal
End With
```



↳ [Show All](#)

PreferredWidth Property

▶ [PreferredWidth property as it applies to the Cell, Cells, Column, Columns, and Table objects.](#)

Returns or sets the preferred width (in points or as a percentage of the window width) for the specified cell, cells, columns, or table. Read/write **Single**.

expression.**PreferredWidth**

expression Required. An expression that returns one of the above objects.

▶ [PreferredWidth property as it applies to the TableStyle object.](#)

Returns or sets the preferred width (in points or as a percentage of the window width) for the specified table style. Read-only **Single**.

expression.**PreferredWidth**

expression Required. An expression that returns one of the above objects.

Remarks

If the [PreferredWidthType](#) property is set to **wdPreferredWidthPoints**, the **PreferredWidth** property returns or sets the width in points. If the **PreferredWidthType** property is set to **wdPreferredWidthPercent**, the **PreferredWidth** property returns or sets the width as a percentage of the window width.

Example

This example sets Microsoft Word to accept preferred widths as a percentage of window width, and then sets the preferred width of the first table in the document to 50% of the window width.

```
With ActiveDocument.Tables(1)
    .PreferredWidthType = wdPreferredWidthPercent
    .PreferredWidth = 50
End With
```



↳ [Show All](#)

PreferredWidthType Property

Returns or sets the preferred unit of measurement to use for the width of the specified cells, columns, or table. Read-only **WdPreferredWidthType** for the [ConditionalStyle](#) and [TableStyle](#) objects; read/write [WdPreferredWidthType](#) for all other objects in the Applies To list.

WdPreferredWidthType can be one of these WdPreferredWidthType constants.

wdPreferredWidthAuto

wdPreferredWidthPercent

wdPreferredWidthPoints

expression.**PreferredWidthType**

expression Required. An expression that returns one of the above objects.

Example

This example sets Microsoft Word to accept widths as a percentage of window width, and then it sets the width of the first table in the document to 50% of the window width.

```
With ActiveDocument.Tables(1)  
    .PreferredWidthType = wdPreferredWidthPercent  
    .PreferredWidth = 50  
End With
```



PreserveFormattingOnUpdate Property

True preserves formatting done in Microsoft Word to a linked OLE object, such as a table linked to a Microsoft Excel spreadsheet. Read/write **Boolean**.

expression.**PreserveFormattingOnUpdate**

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

Remarks

When **PreserveFormattingOnUpdate** is set to **True**, formatting changes made to the object in Word is preserved when the object is updated. Word updates only the content in the linked object.

Example

This example preserves the formatting of the first shape in the current document, assuming the first shape in the document is a linked OLE object.

```
Sub PreserveFmtg()  
    ThisDocument.Shapes(1).OLEFormat _  
        .PreserveFormattingOnUpdate = True  
End Sub
```



↳ [Show All](#)

PresetExtrusionDirection Property

Returns the direction taken by the extrusion's sweep path leading away from the extruded shape (the front face of the extrusion). Read/write [MsoPresetExtrusionDirection](#).

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 a [ThreeDFormat](#) object.

Remarks

This property is read-only. To set the value of this property, use the [SetExtrusionDirection](#) method.

Example

This example changes each extrusion on myDocument that extends toward the upper-left corner of the extrusion's front face to an extrusion that extends toward the lower-right corner of the front face.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
  With s.ThreeD
    If .PresetExtrusionDirection = msoExtrusionTopLeft Then
      .SetExtrusionDirection msoExtrusionBottomRight
    End If
  End With
Next
```



↳ [Show All](#)

PresetGradientType Property

Returns the preset gradient type for the specified fill. Read-only [MsoPresetGradientType](#).

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 a [FillFormat](#) object.

Remarks

Use the [PresetGradient](#) method to set the preset gradient type for the fill.

Example

This example changes the fill for all shapes in myDocument with the Moss preset gradient fill to the Fog preset gradient fill.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
  With s.Fill
    If .PresetGradientType = msoGradientMoss Then
      .PresetGradient msoGradientHorizontal, 1, _
        msoGradientFog
    End If
  End With
Next
```



↳ [Show All](#)

PresetLightingDirection Property

Returns or sets the position of the light source relative to the extrusion.
Read/write [MsoPresetLightingDirection](#).

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 a [ThreeDFormat](#) object.

Remarks

The lighting effects you set won't be apparent if the extrusion has a wire frame surface.

Example

This example specifies that the extrusion for shape one on myDocument extend toward the top of the shape and that the lighting for the extrusion come from the left.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```



↳ [Show All](#)

PresetLightingSoftness Property

Returns or sets the intensity of the extrusion lighting. Read/write [MsoPresetLightingSoftness](#).

MsoPresetLightingSoftness can be one of these MsoPresetLightingSoftness constants.

msoLightingBright

msoLightingDim

msoLightingNormal

msoPresetLightingSoftnessMixed

expression.PresetLightingSoftness

expression Required. An expression that returns a [ThreeDFormat](#) object.

Example

This example specifies that the extrusion for shape one on myDocument be lit brightly from the left.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetLightingSoftness = msoLightingBright
    .PresetLightingDirection = msoLightingLeft
End With
```



↳ [Show All](#)

PresetMaterial Property

Returns or sets the extrusion surface material. Read/write [MsoPresetMaterial](#).

MsoPresetMaterial can be one of these MsoPresetMaterial constants.

msoMaterialMatte

msoMaterialMetal

msoMaterialPlastic

msoMaterialWireFrame

msoPresetMaterialMixed

expression.**PresetMaterial**

expression Required. An expression that returns a [ThreeDFormat](#) object.

Example

This example specifies that the extrusion surface for shape one in myDocument be wire frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetMaterial = msoMaterialWireFrame
End With
```



↳ [Show All](#)

PresetShape Property

Returns or sets the shape of the specified WordArt. Read/write [MsoPresetTextEffectShape](#).

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 a [TextEffectFormat](#) object.

Remarks

Setting the [PresetTextEffect](#) property automatically sets the **PresetShape** property.

Example

This example sets the shape of all WordArt on myDocument to a chevron whose center points down.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
    If s.Type = msoTextEffect Then
        s.TextEffect.PresetShape = msoTextEffectShapeChevronDown
    End If
Next
```



↳ [Show All](#)

PresetTextEffect Property

Returns or sets the style of the specified WordArt. The values for this property correspond to the formats in the **WordArt Gallery** dialog box (**Insert** menu), numbered from left to right, top to bottom. Read/write [MsoPresetTextEffect](#).

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 a [TextEffectFormat](#) object.

Remarks

Setting the **PresetTextEffect** property automatically sets many other formatting properties of the specified shape.

Example

This example sets the style for all WordArt on myDocument to the first style listed in the **WordArt Gallery** dialog box.

```
Set myDocument = ActiveDocument
For Each s In myDocument.Shapes
    If s.Type = msoTextEffect Then
        s.TextEffect.PresetTextEffect = msoTextEffect1
    End If
Next
```



↳ [Show All](#)

PresetTexture Property

Returns the preset texture for the specified fill. Read-only [MsoPresetTexture](#).

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 a [FillFormat](#) object.

Remarks

Use the [PresetTextured](#) method to specify the preset texture for the fill.

Example

This example adds a rectangle to myDocument and sets its preset texture to match that of shape two. For the example to work, shape two must have a preset textured fill.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    presetTexture2 = .Item(2).Fill.PresetTexture
    .AddShape(msoShapeRectangle, 100, 0, 40, 80).Fill _
        .PresetTextured presetTexture2
End With
```



↳ [Show All](#)

PresetThreeDFormat Property

Returns 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](#).

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 a [ThreeDFormat](#) object.

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** button on the **Drawing** toolbar.

Use the [SetThreeDFormat](#) method to set the preset extrusion format.

Example

This example sets the extrusion format for shape one on myDocument to 3-D Style 12 if the shape initially has a custom extrusion format.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    If .PresetThreeDFormat = msoPresetThreeDFormatMixed Then
        .SetThreeDFormat msoThreeD12
    End If
End With
```



Previous Property

-

Returns the previous object in the collection. Read-only.

Example

This example sets the space-before and space-after formatting for the paragraph immediately preceding the selection.

```
Set myPara = Selection.Paragraphs(1).Previous
With myPara
    .SpaceAfter = 12
    .SpaceBefore = 6
End With
```

If the selection is in a table, this example selects the contents of the previous row.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Rows(1).Previous.Select
End If
```

This example displays the field code of the second-to-last field in the active document.

```
Set aField = ActiveDocument _
    .Fields(ActiveDocument.Fields.Count).Previous
MsgBox "Field code = " & aField.Code
```



PreviousBookmarkID Property

-

Returns the number of the last bookmark that starts before or at the same place as the specified selection or range; returns 0 (zero) if there's no corresponding bookmark. Read-only **Long**.

Example

This example selects the previous bookmark in the active document.

```
num = Selection.PreviousBookmarkID  
If num <> 0 Then ActiveDocument.Content.Bookmarks(num).Select
```

This example displays the name of the bookmark that precedes the second paragraph.

```
num = ActiveDocument.Paragraphs(2).Range.PreviousBookmarkID  
If num <> 0 Then MsgBox ActiveDocument.Content.Bookmarks(num).Name
```



PrintBackground Property

True if Microsoft Word prints in the background. Read/write **Boolean**.

expression.**PrintBackground**

expression Required. An expression that returns an [Options](#) object.

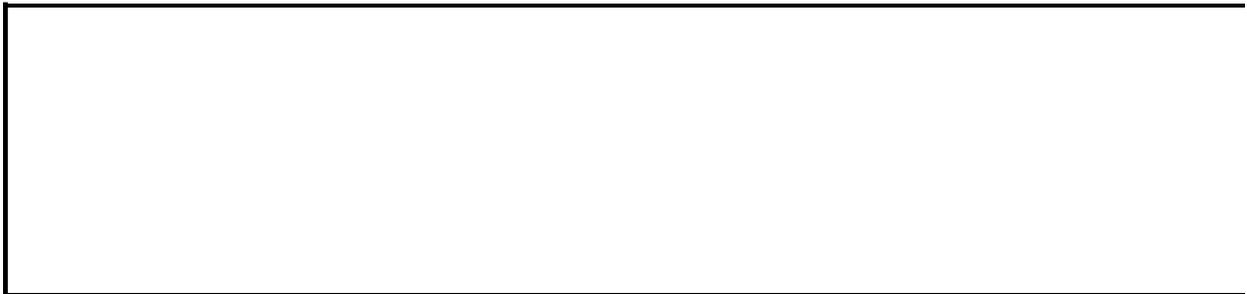
Example

This example sets Word to print documents in the background and then prints the active document.

```
Options.PrintBackground = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Background printing** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintBackground
```



PrintComments Property

True if Microsoft Word prints comments, starting on a new page at the end of the document. Read/write **Boolean**.

expression.**PrintComments**

expression Required. An expression that returns an [Options](#) object.

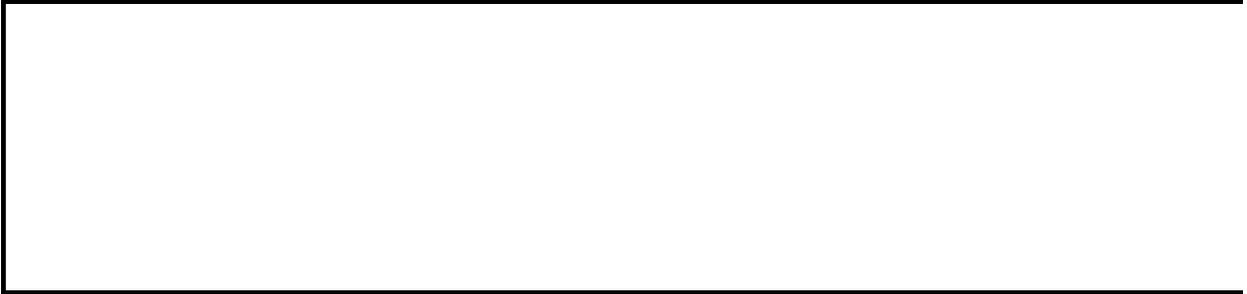
Remarks

Setting the **PrintComments** property to **True** automatically sets the **PrintHiddenText** property to **True**. However, setting the **PrintComments** property to **False** has no effect on the setting of the **PrintHiddenText** property.

Example

This example sets Word to print comments and then prints the active document.

```
Options.PrintComments = True  
ActiveDocument.PrintOut
```



PrintDraft Property

True if Microsoft Word prints using minimal formatting. Read/write **Boolean**.

expression.**PrintDraft**

expression Required. An expression that returns an [Options](#) object.

Remarks

Not all printers support draft printing.

Example

This example sets Word to use draft printing and then prints the active document.

```
Options.PrintDraft = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Draft output** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintDraft
```



PrintDrawingObjects Property

True if Microsoft Word prints drawing objects. Read/write **Boolean**.

expression.**PrintDrawingObjects**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to print drawing objects, and then it prints the active document.

```
Options.PrintDrawingObjects = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Drawing objects** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintDrawingObjects
```



PrintEvenPagesInAscendingOrder Property

True if Microsoft Word prints even pages in ascending order during manual duplex printing. Read/write **Boolean**.

expression.**PrintEvenPagesInAscendingOrder**

expression Required. An expression that returns an [Options](#) object.

Remarks

If the *ManualDuplexPrint* argument of the [PrintOut](#) method is **False**, this property is ignored.

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Word to print odd pages in ascending order and even pages in descending order during manual duplex printing, and then it prints the active document.

```
Options.PrintOddPagesInAscendingOrder = True  
Options.PrintEvenPagesInAscendingOrder = False  
ActiveDocument.PrintOut ManualDuplexPrint:=True
```



PrintFieldCodes Property

True if Microsoft Word prints field codes instead of field results. Read/write **Boolean**.

expression.**PrintFieldCodes**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to print field codes, and then it prints the active document.

```
Options.PrintFieldCodes = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Field codes** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintFieldCodes
```



PrintFormsData Property

-

True if Microsoft Word prints onto a preprinted form only the data entered in the corresponding online form. Read/write **Boolean**.

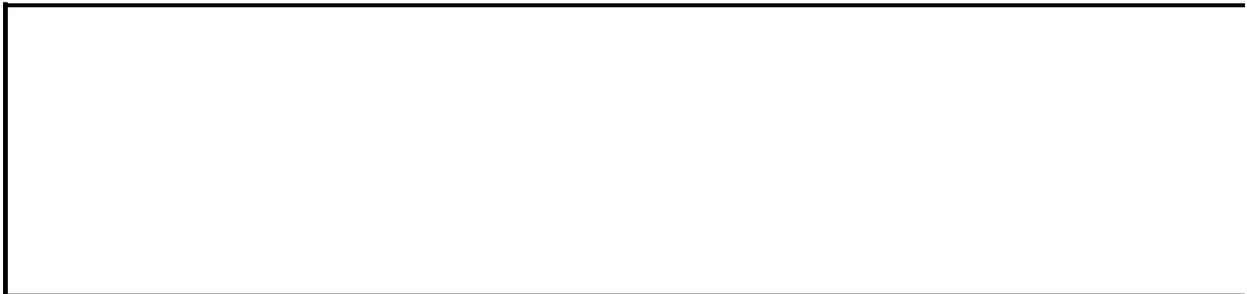
Example

This example sets Word to print only the data from an online form, and then it prints the active document.

```
ActiveDocument.PrintFormsData = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Print data only for forms** check box in the **Options for current document only** area on the **Print** tab in the **Options** dialog box.

```
temp = ActiveDocument.PrintFormsData
```



PrintFractionalWidths Property

True if the specified document is formatted to use fractional point spacing to display and print characters. Read/write **Boolean**.

Note In Windows, this property always returns **False**. For additional information about this property, consult the language reference Help included with Microsoft Office Macintosh Edition.



PrintHiddenText Property

True if hidden text is printed. Read/write **Boolean**.

expression.**PrintHiddenText**

expression Required. An expression that returns an [Options](#) object.

Remarks

Setting the **PrintHiddenText** property to **False** automatically sets the [PrintComments](#) property to **False**. However, setting the **PrintHiddenText** property to **True** has no effect on the setting of the **PrintComments** property.

Example

This example sets Word to print hidden text, and then it prints the active document.

```
Options.PrintHiddenText = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Hidden text** option on the **Print** tab in the **Options** dialog box.

```
temp = Options.PrintHiddenText
```



PrintOddPagesInAscendingOrder Property

True if Microsoft Word prints odd pages in ascending order during manual duplex printing. Read/write **Boolean**.

expression.**PrintOddPagesInAscendingOrder**

expression Required. An expression that returns an [Options](#) object.

Remarks

If the *ManualDuplexPrint* argument of the [PrintOut](#) method is **False**, this property is ignored.

For more information on using Word with East Asian languages, see [Word features for East Asian languages](#).

Example

This example sets Microsoft Word to print odd pages in ascending order and even pages in descending order during manual duplex printing, and then it prints the active document.

```
Options.PrintOddPagesInAscendingOrder = True  
Options.PrintEvenPagesInAscendingOrder = False  
ActiveDocument.PrintOut ManualDuplexPrint:=True
```



PrintPostScriptOverText Property

-

True if PRINT field instructions (such as PostScript commands) in a document are to be printed on top of text and graphics when a PostScript printer is used. Read/write **Boolean**.

Remarks

This property controls whether postscript code is printed in a converted Microsoft Word for Macintosh document. If the document contains no PRINT fields, this property has no effect.

Example

This example sets Word to print PRINT field instructions on top of text and graphics, and then it prints the active document.

```
ActiveDocument.PrintPostScriptOverText = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Print PostScript over text** check box in the **Printing options** area on the **Print** tab in the **Options** dialog box.

```
currSet = ActiveDocument.PrintPostScriptOverText
```



PrintPreview Property

True if print preview is 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.

```
PrintPreview = True
```

This example switches the active window from print preview to normal view.

```
PrintPreview = False
```

```
ActiveDocument.ActiveWindow.View.Type = wdNormalView
```



PrintProperties Property

-

True if Microsoft Word prints document summary information on a separate page at the end of the document. **False** if document summary information is not printed. Summary information is found in the **Properties** dialog box (**File** menu). Read/write **Boolean**.

expression.**PrintProperties**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to print document summary information on a separate page at the end of the document, and then it prints the active document.

```
Options.PrintProperties = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Document properties** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintProperties
```



PrintReverse Property

True if Microsoft Word prints pages in reverse order. Read/write **Boolean**.

expression.**PrintReverse**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to print pages in reverse order, and then it prints the active document.

```
Options.PrintReverse = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Reverse print order** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.PrintReverse
```



PrintRevisions Property

-

True if revision marks are printed with the document. **False** if revision marks aren't printed (that is, tracked changes are printed as if they'd been accepted).
Read/write **Boolean**.

Example

This example prints the active document without revision marks.

```
With ActiveDocument  
    .PrintRevisions = False  
    .PrintOut  
End With
```



PrivateProfileString Property

Returns or sets a string in a settings file or the Windows registry. Read/write **String**.

expression.**PrivateProfileString**(*FileName*, *Section*, *Key*)

expression Required. An expression that returns a [System](#) object.

FileName Required **String**. The file name for the settings file. If there's no path specified, the Windows folder is assumed. If you're using Windows 95, Windows 98, or Windows NT to return the value of a registry entry, **FileName** must be an empty string ("").

Section Required **String**. The name of the section in the settings file that contains **Key**. In a Windows settings file, the section name appears between brackets before the associated keys (don't include the brackets with **Section**). If you're returning the value of an entry from the Windows registry, **Section** should be the complete path to the subkey, including the subtree (for example, "HKEY_CURRENT_USER\Software\Microsoft\Office*version*\Word\Options").

Key Required **String**. The key setting or registry entry value you want to retrieve. In a Windows settings file, the key name is followed by an equal sign (=) and the setting. If you're returning the value of an entry from the Windows registry, **Key** should be the name of an entry in the subkey specified by **Section** (for example, "STARTUP-PATH").

Remarks

You can write macros that use a settings file to store and retrieve settings. For example, you can store the name of the active document when you quit Word so that it can be reopened automatically the next time you start Word. A settings file is a text file with information arranged like the information in the Windows 3.x WIN.INI file.

Example

This example sets the current document name as the LastFile setting under the MacroSettings heading in Settings.txt.

```
System.PrivateProfileString("C:\Settings.txt", "MacroSettings", _  
    "LastFile") = ActiveDocument.FullName
```

This example returns the LastFile setting from Settings.txt and then opens the document stored in LastFile.

```
LastFile = System.PrivateProfileString("C:\Settings.Txt", _  
    "MacroSettings", "LastFile")  
If LastFile <> "" Then Documents.Open FileName:=LastFile
```

This example displays the value of the EmailName entry from the Windows registry.

```
aName = System.PrivateProfileString("", _  
    "HKEY_CURRENT_USER\Software\Microsoft\" _  
    & "Windows\CurrentVersion\Internet Settings", "EmailName")  
MsgBox aName
```



ProcessorType Property

Returns the type of processor that the system is using (for example, i486). Read-only **String**.

expression.**ProcessorType**

expression Required. An expression that returns a [System](#) object.

Example

This example displays a message on the status bar if the processor that the system is using isn't a Pentium processor.

```
If System.ProcessorType <> "Pentium" Then _  
    StatusBar = "Please wait..."
```



ProfileString Property

Returns or sets a value for an entry in the Windows registry under the following subkey: HKEY_CURRENT_USER\Software\Microsoft\Office*version*\Word.
Read/write **String**.

expression.**ProfileString**(*Section*, *Key*)

expression Required. An expression that returns a [System](#) object.

Section Required **String**. A subkey below the "HKEY_CURRENT_USER\Software\Microsoft\Office*version*\Word" subkey in the Windows registry.

Key Required **String**. The name of the entry in the subkey specified by **Section** (for example, "BackgroundPrint" in the Options subkey).

Example

This example retrieves and displays the startup path stored in the Windows registry.

```
MsgBox System.ProfileString("Options", "STARTUP-PATH")
```

This example sets and returns the value for an entry in the Windows registry (the SubkeyName subkey is added below

HKEY_CURRENT_USER\Software\Microsoft\Office*version*\Word).

```
System.ProfileString("SubkeyName", "EntryName") = "Value"  
MsgBox System.ProfileString("SubkeyName", "EntryName")
```



↳ [Show All](#)

ProgID Property

Returns the [programmatic identifier \(ProgID\)](#) for the specified OLE object.
Read-only **String**.

expression.**ProgID**

expression Required. An expression that returns an [OLEFormat](#) object.

Remarks

The **ProgID** and [ClassType](#) properties will (by default) return the same string. However, you can change the **ClassType** property for DDE links.

For information about programmatic identifiers, see [OLE Programmatic Identifiers](#).

Example

This example loops through all the floating shapes in the active document and sets all linked Microsoft Excel worksheets to be updated automatically.

```
For Each s In ActiveDocument.Shapes
    If s.Type = msoLinkedOLEObject Then
        If s.OLEFormat.ProgID = "Excel.Sheet" Then
            s.LinkFormat.AutoUpdate = True
        End If
    End If
Next
```



PromptUpdateStyle Property

True displays a message asking the user to verify whether they want to reformat a style or reapply the original style formatting when changing the formatting of styles. **False** reapplies the style formatting to the selection without verifying whether the user wants to change the style. Read/write **Boolean**.

expression.**PromptUpdateStyle**

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

Example

This example checks to see if a user receives a message when updating styles, and if not, enables it.

```
Sub UpdateStylePrompt()  
  With Application.Options  
    If .PromptUpdateStyle = False Then  
      .PromptUpdateStyle = True  
    End If  
  End With  
End Sub
```



Properties Property

Returns a [CustomProperties](#) object that represents the properties of a smart tag.

expression.**Properties**

expression Required. An expression that returns a [SmartTag](#) object.

Remarks

You can use the [Add](#) method to add custom properties from within a Microsoft Word Visual Basic for Applications project. However, custom properties are generally specified in the smart tag recognizer and action files.

Example

This example loops through all the smart tags in the current document, and then it creates a new document and lists the names and values of custom properties for all smart tags that have custom properties.

```
Sub SmartTagProps()  
    Dim docNew As Document  
    Dim stgTag As SmartTag  
    Dim stgProp As CustomProperty  
    Dim intTag As Integer  
    Dim intProp As Integer  
  
    'Create new document and add heading content  
    Set docNew = Documents.Add  
  
    With docNew.Content  
        .InsertAfter "Name" & vbTab & "Value"  
        .InsertParagraphAfter  
    End With  
  
    'Loop through smart tags in current document  
    For intTag = 1 To ThisDocument.SmartTags.Count  
  
        With ThisDocument.SmartTags(intTag)  
  
            'Verify that a smart tag has properties  
            If .Properties.Count > 0 Then  
  
                'Enter the name and value of properties into new doc  
                For intProp = 1 To .Properties.Count  
                    docNew.Content.InsertAfter .Properties(intProp)  
                        .Name & vbTab & .Properties(intProp).Value  
                    docNew.Content.InsertParagraphAfter  
                Next  
            Else  
  
                'Display message if no properties for smart tag  
                MsgBox "There are no custom properties for this smar  
            End If  
        End With  
    Next  
  
    'Convert the tabbed list in the new document to a table
```

```
docNew.Content.Select  
Selection.ConvertToTable Separator:=wdSeparateByTabs, NumColumns
```

```
End Sub
```



↳ [Show All](#)

Protect Property

Returns or sets the protection type for the document associated with the specified routing slip. Read/write [WdProtectionType](#).

WdProtectionType can be one of these WdProtectionType constants.

wdAllowOnlyComments

wdAllowOnlyFormFields

wdAllowOnlyRevisions

wdNoProtection

expression.**Protect**

expression Required. An expression that returns a [RoutingSlip](#) object.

Example

This example protects the active document (only allows comments) and then routes it.

```
ActiveDocument.HasRoutingSlip = True
With ActiveDocument.RoutingSlip
    .Subject = "Status Doc"
    .Protect = wdAllowOnlyComments
    .AddRecipient Recipient:="Kim Johnson"
End With
ActiveDocument.Route
```



Protected Property

-
True if you cannot change the specified key binding in the **Customize Keyboard** dialog box (from the **Tools** menu, click **Customize**, and then click the **Keyboard** button). Read-only **Boolean**.

expression.**Protected**

expression Required. An expression that returns a [KeyBinding](#) object.

Remarks

Use the [Add](#) method of the [KeyBindings](#) object to add a key binding regardless of the protected status.

Example

This example displays the protection status for the CTRL+S key binding.

```
CustomizationContext = ActiveDocument.AttachedTemplate  
MsgBox FindKey(BuildKeyCode(wdKeyControl, wdKeyS)).Protected
```

This example displays a message if the A key binding is protected.

```
CustomizationContext = NormalTemplate  
If FindKey(BuildKeyCode(wdKeyA)).Protected = True Then  
    MsgBox "The A key is protected"  
End If
```



ProtectedForForms Property

-
True if the specified section is protected for forms. When a section is protected for forms, you can select and modify text only in form fields. Read/write **Boolean**.

expression.**ProtectedForForms**

expression Required. An expression that returns a [Section](#) object.

Remarks

To protect an entire document, use the [Protect](#) method of the [Document](#) object.

Example

This example protects the second section in the active document for forms.

```
If ActiveDocument.Sections.Count >= 2 Then _  
    ActiveDocument.Sections(2).ProtectedForForms = True
```

This example unprotects the first section in the selection.

```
Selection.Sections(1).ProtectedForForms = False
```

This example toggles the protection for the first section in the selection.

```
Selection.Sections(1).ProtectedForForms = Not _  
    Selection.Sections(1).ProtectedForForms
```



ProtectionType Property

-

Returns the protection type for the specified document. Can be one of the following **WdProtectionType** constants: **wdAllowOnlyComments**, **wdAllowOnlyFormFields**, **wdAllowOnlyRevisions**, or **wdNoProtection**.
Read-only **Long**.

Example

If the active document isn't already protected, this example protects the document for comments.

```
If ActiveDocument.ProtectionType = wdNoProtection Then  
    ActiveDocument.Protect Type:=wdAllowOnlyComments  
End If
```

This example unprotects the active document if it's protected.

```
Set Doc = ActiveDocument  
If Doc.ProtectionType <> wdNoProtection Then Doc.Unprotect
```



QueryString Property

Returns or sets the query string (SQL statement) used to retrieve a subset of the data in a mail merge data source. Read/write **String**.

expression.**QueryString**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

Example

This example returns the query string for the data source attached to the active document.

```
qString = ActiveDocument.MailMerge.DataSource.QueryString
```



Range Property

Returns a [Range](#) object that represents the portion of a document that's contained in the specified object.

expression.**Range**

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

Remarks

For information about returning a range from a document or returning a shape range from a collection of shapes, see the [Range](#) method.

Example

This example applies the Heading 1 style to the first paragraph in the active document.

```
ActiveDocument.Paragraphs(1).Range.Style = wdStyleHeading1
```

This example copies the first row in table one.

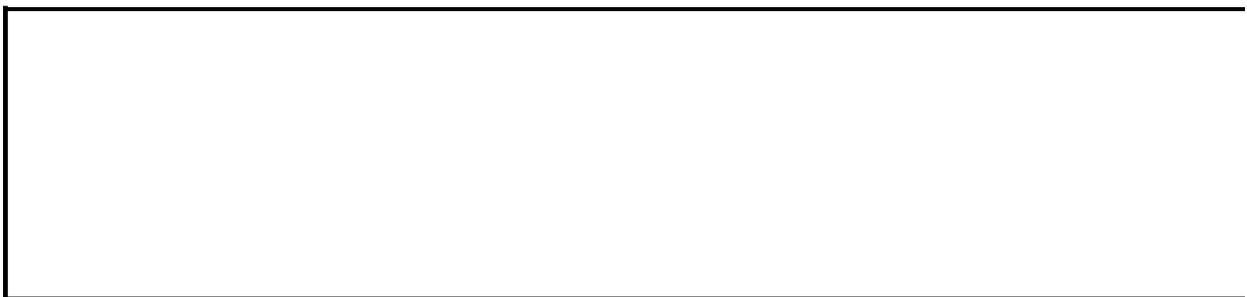
```
If ActiveDocument.Tables.Count >= 1 Then _  
    ActiveDocument.Tables(1).Rows(1).Range.Copy
```

This example changes the text of the first comment in the document.

```
With ActiveDocument.Comments(1).Range  
    .Delete  
    .InsertBefore "new comment text"  
End With
```

This example inserts text at the end of section one.

```
Set myRange = ActiveDocument.Sections(1).Range  
With myRange  
    .MoveEnd Unit:=wdCharacter, Count:=-1  
    .Collapse Direction:=wdCollapseEnd  
    .InsertParagraphAfter  
    .InsertAfter "End of section"  
End With
```



ReadabilityStatistics Property

Returns a [ReadabilityStatistics](#) collection that represents the readability statistics for the specified document or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays each readability statistic, along with its value, for document one.

```
For Each rs In Documents(1).ReadabilityStatistics  
    MsgBox rs.Name & " - " & rs.Value  
Next rs
```



↳ [Show All](#)

ReadingOrder Property

Returns or sets the reading order of the specified paragraphs without changing their alignment. Read/write [WdReadingOrder](#).

WdReadingOrder can be one of these WdReadingOrder constants.

wdReadingOrderLtr

wdReadingOrderRtl

expression.**ReadingOrder**

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

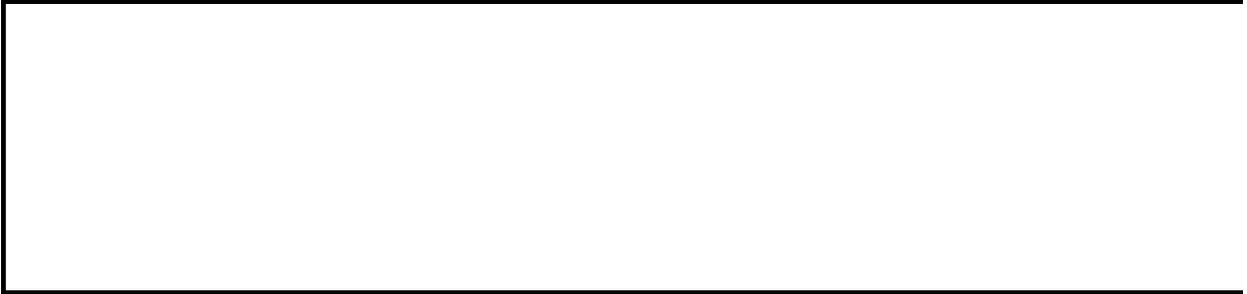
Remarks

Use the [LtrPara](#), [LtrRun](#), [RtlPara](#), and [RtlRun](#) methods to change the paragraph alignment along with the reading order.

Example

This example sets the reading order of the first paragraph to right-to-left.

```
ActiveDocument.Paragraphs(1).ReadingOrder = _  
    wdReadingOrderRtl
```



↳ [Show All](#)

ReadOnly Property

▶ [ReadOnly property as it applies to the **Dictionary** and **Document** objects.](#)

Dictionary object: **True** if the specified dictionary cannot be changed. Read-only **Boolean**.

Document object: **True** if changes to the document cannot be saved to the original document. Read-only **Boolean**.

expression.**ReadOnly**

expression Required. An expression that returns one of the above objects.

Note The active grammar, hyphenation, spelling, and thesaurus dictionaries are read-only. Custom dictionaries are read/write.

▶ [ReadOnly property as it applies to the **RecentFile** object.](#)

True if changes to the document cannot be saved to the original document. Read/write **Boolean**.

expression.**ReadOnly**

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

Example

▶ [As it applies to the **Dictionary** and **Document** objects.](#)

This example saves the active document if it isn't read-only.

```
If ActiveDocument.ReadOnly = False Then ActiveDocument.Save
```

▶ [As it applies to the **RecentFile** object.](#)

This example opens the most recently used file as a read-only document.

```
With RecentFiles(1)  
    .ReadOnly = True  
    .Open  
End With
```



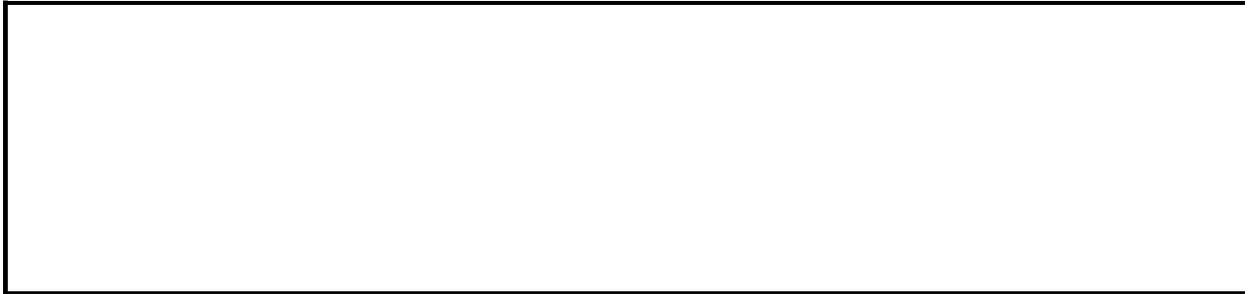
ReadOnlyRecommended Property

-
True if Word displays a message box whenever a user opens the document, suggesting that it be opened as read-only. Read/write **Boolean**.

Example

This example sets Word to suggest, when it's opening the document, that the document be opened as read-only.

```
ActiveDocument.ReadOnlyRecommended = True
```



RecentFiles Property

Returns a [RecentFiles](#) collection that represents the most recently accessed files.

expression.**RecentFiles**

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 opens the first item in the **RecentFiles** collection (the first document name listed on the **File** menu).

```
If RecentFiles.Count >= 1 Then RecentFiles(1).Open
```

This example displays the name of each file in the **RecentFiles** collection.

```
For Each rFile In RecentFiles  
    MsgBox rFile.Name  
Next rFile
```



RecipientAddress Property

Returns or sets the mailing address of the person who'll be receiving the letter created by the Letter Wizard. Read/write **String**.

expression.**RecipientAddress**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, sets several properties (including the recipient address), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim oLC as New LetterContent
With oLC
    .ReturnAddress = Application.UserAddress
    .RecipientName = "Amy Anderson"
    .RecipientAddress = "123 Main" & vbCr & "Bellevue, WA 98004"
End With
Documents.Add.RunLetterWizard LetterContent:=oLC, WizardMode:=True
```



RecipientCode Property

Returns or sets the recipient code. Not used in the U.S. English version of Microsoft Word. Read/write **String**.

expression.**RecipientCode**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

↳ [Show All](#)

RecipientGender Property

-
Returns or sets the recipient's gender, if known. Not used in the U.S. English version of Microsoft Word. Read/write [WdSalutationGender](#).

WdSalutationGender can be one of these WdSalutationGender constants.

wdGenderFemale

wdGenderMale

wdGenderNeutral

wdGenderUnknown

expression.**RecipientGender**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

RecipientName Property

Returns or sets the name of the person who'll be receiving the letter created by the Letter Wizard. Read/write **String**.

expression.**RecipientName**

expression Required. An expression that returns a [LetterContent](#) object.

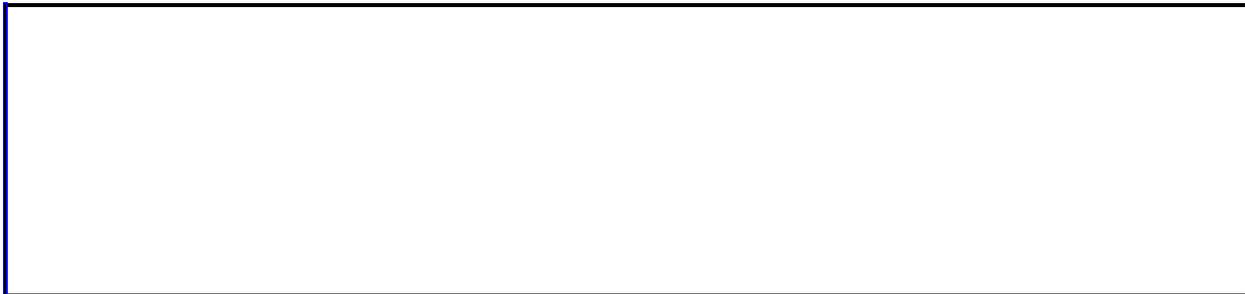
Example

This example displays the salutation and recipient name for the active document.

```
MsgBox ActiveDocument.GetLetterContent.Salutation _  
    & Space(1) & ActiveDocument.GetLetterContent.RecipientName
```

This example creates a new [LetterContent](#) object, sets several properties (including the recipient name), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim oLC as New LetterContent  
With oLC  
    .LetterStyle = wdFullBlock  
    .ReturnAddress = Application.UserAddress  
    .RecipientName = "Amy Anderson"  
    .RecipientAddress = "123 Main" & vbCrLf & "Bellevue, WA 98004"  
End With  
Documents.Add.RunLetterWizard LetterContent:=oLC, WizardMode:=True
```



RecipientNamefromLeft Property

Returns or sets a **Single** that represents the position, measured in points, of the recipient's name from the left edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**RecipientNamefromLeft**

expression Required. An expression that returns an [Envelope](#) object.

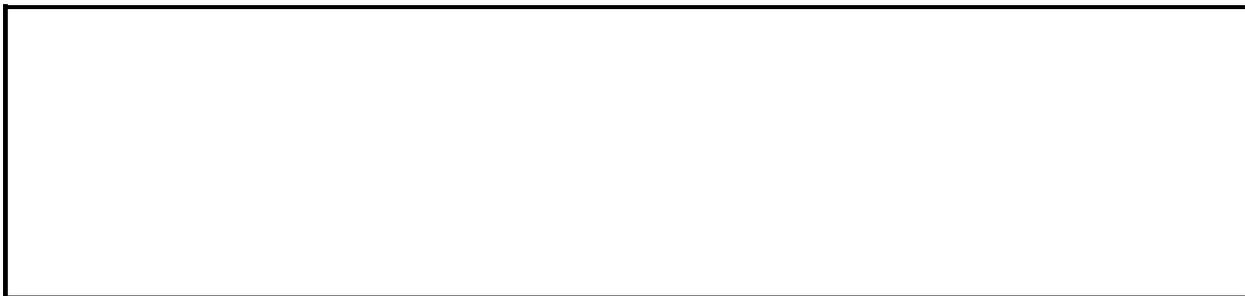
Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



RecipientNamefromTop Property

Returns or sets a **Single** that represents the position, measured in points, of the recipient's name from the top edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**RecipientNamefromTop**

expression Required. An expression that returns an [Envelope](#) object.

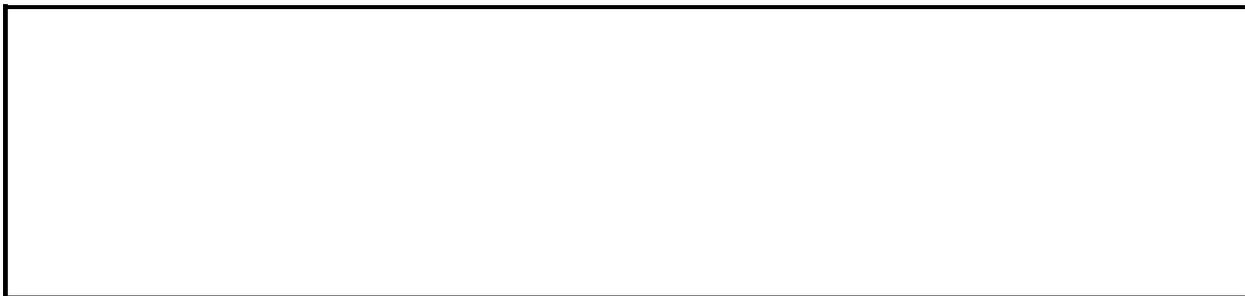
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



RecipientPostalfromLeft Property

Returns or sets a **Single** that represents the position, measured in points, of the recipient's postal code from the left edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**RecipientPostalfromLeft**

expression Required. An expression that returns an [Envelope](#) object.

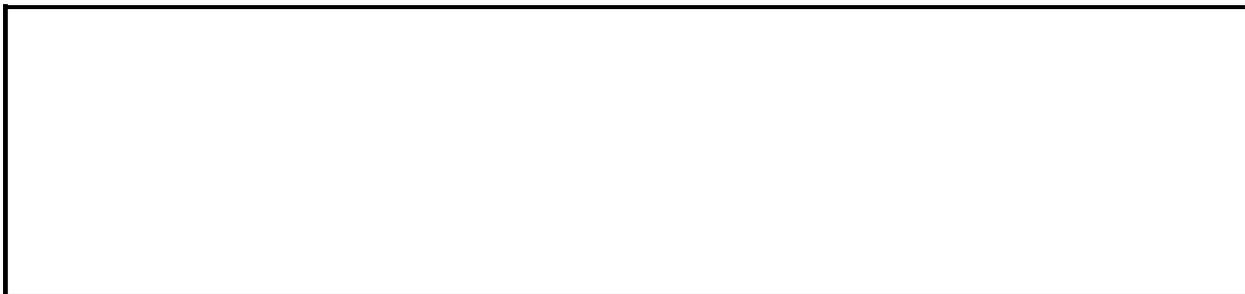
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



RecipientPostalfromTop Property

Returns or sets a **Single** that represents the position, measured in points, of the recipient's postal code from the top edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**RecipientPostalfromTop**

expression Required. An expression that returns an [Envelope](#) object.

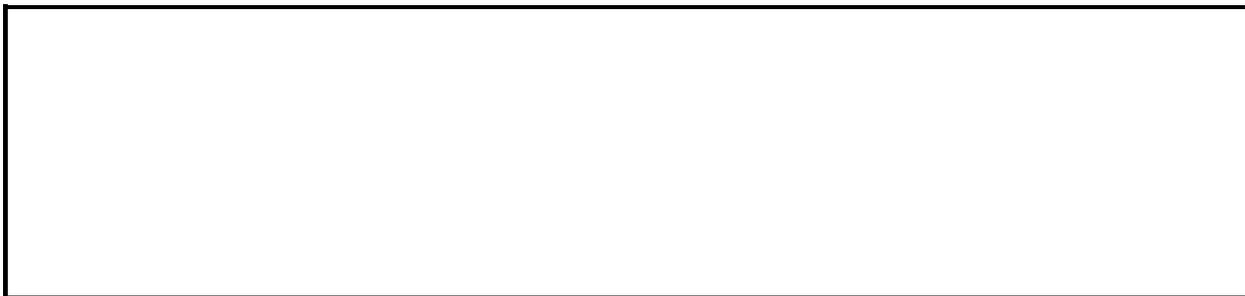
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



RecipientReference Property

Returns or sets the reference line (for example, "In reply to:") for a letter created by the Letter Wizard. Read/write **String**.

expression.**RecipientReference**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, sets several properties (including the reference line), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
With myContent
    .RecipientReference = "In reply to:"
    .Salutation = "Hello"
    .MailingInstructions = "Certified Mail"
End With
Documents.Add.RunLetterWizard LetterContent:=myContent
```



Recipients Property

Returns a recipient name from the specified routing slip. Read-only **Variant**.

expression.**Recipients**(*Index*)

expression Required. An expression that returns a [RoutingSlip](#) object.

Index Optional **Variant**. A number that specifies the recipient (in the list of recipients).

Example

This example adds a recipient to the routing slip attached to Sales.doc and then displays the name of the first recipient.

```
If Documents("Sales.doc").HasRoutingSlip = True Then
    Documents("Sales.doc").RoutingSlip.AddRecipient _
        Recipient:="Aaron Con"
    MsgBox Documents("Sales.doc").RoutingSlip.Recipients(1)
End If
```



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.

Remarks

If Microsoft Word cannot determine the number of records in a data source, the **RecordCount** property will return a value of -1.

Example

This example loops through the records in the data source and verifies that the postal code field (field six in this example) is not less than five digits. If it is, it removes the record from the mail merge. If you want to make sure that the locator code is added to the postal code, you can change the length value from 5 to 10. Therefore, if a postal code is less than ten digits it will be removed from the mail merge.

```
Sub ExcludeRecords()  
  
    On Error GoTo ErrorHandler  
  
    With ActiveDocument.MailMerge.DataSource  
        .ActiveRecord = wdFirstRecord  
        Do  
  
            'Counts the number of digits in the postal code field an  
            'it is less than 5, the record is excluded from the mail  
            'marked as having an invalid address, and given a commen  
            'describing why the postal code was removed  
            If Len(.DataFields(6).Value) < 5 Then  
                .Included = False  
                .InvalidAddress = True  
                .InvalidComments = "The zip code for this record" &  
                    "is less than five digits. This record is" & _  
                    "removed from the mail merge process."  
            End If  
            If .ActiveRecord <> .RecordCount Then  
                .ActiveRecord = wdNextRecord  
            End If  
        Loop Until .ActiveRecord = .RecordCount  
    ErrorHandler:  
  
    End With  
  
End Sub
```



Reference Property

Returns a [Range](#) object that represents a footnote, endnote, or comment reference mark.

expression.**Reference**

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

Example

This example sets myRange to the first footnote reference mark in the active document and then copies the reference mark.

```
Set myRange = ActiveDocument.Footnotes(1).Reference  
myRange.Copy
```

This example formats the comment reference marks in the selection to be red.

```
For Each comm In Selection.Comments  
    comm.Reference.Font.ColorIndex = wdRed  
Next comm
```



RelatedExpressionList Property

Returns a list of expressions related to the specified word or phrase. The list is returned as an array of strings. Read-only **Variant**.

expression.**RelatedExpressionList**

expression Required. An expression that returns a [SynonymInfo](#) object.

Remarks

Typically, there are very few related expressions found in the thesaurus.

Example

This example checks to see whether any related expressions were found for the selection. If so, the meanings are displayed in a series of message boxes. If none were found, this is stated in a message box.

```
Set synInfo = Selection.Range.SynonymInfo
If synInfo.Found = True Then
    relList = synInfo.RelatedExpressionList
    If UBound(relList) <> 0 Then
        For intCount = 1 To UBound(relList)
            MsgBox relList(intCount)
        Next intCount
    Else
        MsgBox "There were no related expressions found."
    End If
End If
```



RelatedWordList Property

Returns a list of words related to the specified word or phrase. The list is returned as an array of strings. Read-only **Variant**.

expression.**RelatedWordList**

expression Required. An expression that returns a [SynonymInfo](#) object.

Example

This example checks to see whether any related words were found for the third word in the active document. If so, the meanings are displayed in a series of message boxes. If there are no related words found, this is stated in a message box.

```
Set synInfo = ActiveDocument.Words(3).SynonymInfo
If synInfo.Found = True Then
    relList = synInfo.RelatedWordList
    If UBound(relList) <> 0 Then
        For intCount = 1 To UBound(relList)
            MsgBox relList(intCount)
        Next intCount
    Else
        MsgBox "There were no related words found."
    End If
End If
```



↳ [Show All](#)

RelativeHorizontalPosition Property

-
Specifies to what the horizontal position of a frame, a shape, or a group of rows is relative. Read/write [WdRelativeHorizontalPosition](#).

Can be one of the following **WdRelativeHorizontalPosition** constants.

wdRelativeHorizontalPositionCharacter

wdRelativeHorizontalPositionColumn

wdRelativeHorizontalPositionMargin

wdRelativeHorizontalPositionPage.

Example

▶ [As it relates to the `wdRelativeHorizontalPositionMargin` constant.](#)

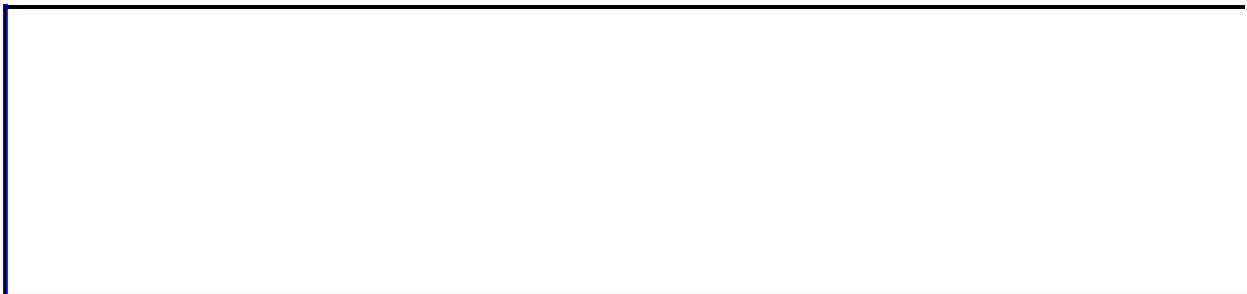
This example adds a frame around the selection and aligns the frame horizontally with the right margin.

```
Set myFrame = ActiveDocument.Frames.Add(Range:=Selection.Range)
With myFrame
    .RelativeHorizontalPosition = _
        wdRelativeHorizontalPositionMargin
    .HorizontalPosition = wdFrameRight
End With
```

▶ [As it relates to the `wdRelativeHorizontalPositionPage` constant.](#)

This example repositions the selected shape object.

```
With Selection.ShapeRange
    .Left = InchesToPoints(0.6)
    .RelativeHorizontalPosition = wdRelativeHorizontalPositionPage
    .Top = InchesToPoints(1)
    .RelativeVerticalPosition = wdRelativeVerticalPositionParagraph
End With
```



↳ [Show All](#)

RelativeVerticalPosition Property

-
Specifies to what the vertical position of a frame, a shape, or a group of rows is relative. Read/write [WdRelativeVerticalPosition](#).

Can be one of the following **WdRelativeVerticalPosition** constants.

wdRelativeVerticalPositionLine

wdRelativeVerticalPositionMargin

wdRelativeVerticalPositionPage

wdRelativeVerticalPositionParagraph.

Example

▶ [As it applies to the **Frames** object.](#)

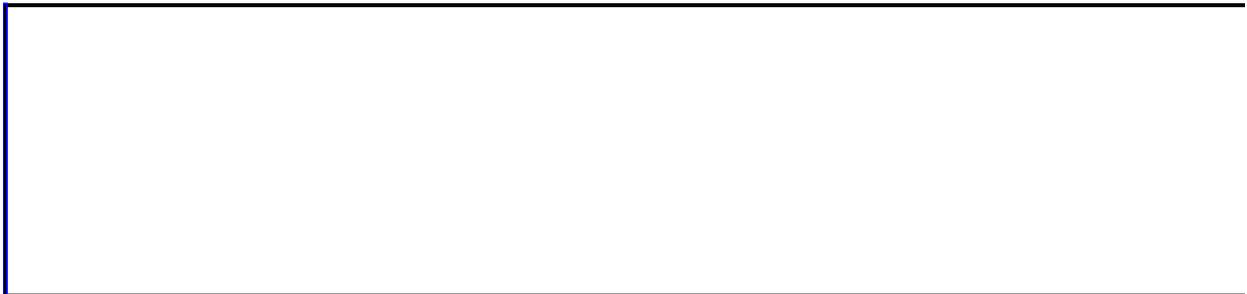
This example adds a frame around the selection and aligns the frame vertically with the top of the page.

```
Set myFrame = ActiveDocument.Frames.Add(Range:=Selection.Range)
With myFrame
    .RelativeVerticalPosition = wdRelativeVerticalPositionPage
    .VerticalPosition = wdFrameTop
End With
```

▶ [As it applies to the **Shape** object.](#)

This example repositions the first shape object in the active document.

```
With ActiveDocument.Shapes(1)
    .Left = InchesToPoints(0.6)
    .RelativeHorizontalPosition = wdRelativeHorizontalPositionPage
    .Top = InchesToPoints(1)
    .RelativeVerticalPosition = wdRelativeVerticalPositionParagraph
End With
```



RelyOnCSS Property

-

True if cascading style sheets (CSS) are used for font formatting when you view a saved document in a Web browser. Microsoft Word creates a cascading style sheet file and saves it either to the specified folder or to the same folder as your Web page, depending on the value of the [OrganizeInFolder](#) property. **False** if HTML tags and cascading style sheets are used. The default value is **True**. Read/write **Boolean**.

expression.**RelyOnCSS**

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

Remarks

You should set this property to **True** if your Web browser supports cascading style sheets because this will give you more precise layout and formatting control on your Web page and make it look more like your document (as it appears in Microsoft Word).

Example

This example enables the use of cascading style sheets as the global default for the application.

```
Application.DefaultWebOptions.RelyOnCSS = True
```



RelyOnVML Property

-
True if image files are not generated from drawing objects when you save a document as a Web page. **False** if images are generated. The default value is **False**. Read/write **Boolean**.

expression.**RelyOnVML**

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

Remarks

You can reduce file sizes by not generating images for drawing objects, if your Web browser supports Vector Markup Language (VML). For example, Microsoft Internet Explorer 5 supports this feature, and you should set the **RelyOnVML** property to **True** if you are targeting this browser. For browsers that do not support VML, the image will not appear when you view a Web page saved with this property enabled.

Don't generate images if your Web page uses image files that you have generated earlier and if the location where you save the document is different from the final location of the page on the Web server.

Example

This example specifies that images are generated when saving the document as a Web page.

```
ActiveDocument.WebOptions.RelOnVML = False
```



RemovePersonalInformation Property

True if Microsoft Word removes all user information from comments, revisions, and the **Properties** dialog box upon saving a document. Read/write **Boolean**.

expression.**RemovePersonalInformation**

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

Example

This example sets the current document to remove personal information from the document the next time the user saves it.

```
Sub RemovePersonalInfo()  
    ThisDocument.RemovePersonalInformation = True  
End Sub
```



Replacement Property

Returns a [Replacement](#) object that contains the criteria for a replace operation.

expression.**Replacement**

expression Required. An expression that returns a [Find](#) object.

Example

This example removes bold formatting from the active document. The **Bold** property is **True** for the **Find** object and **False** for the **Replacement** object.

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Font.Bold = True
    With .Replacement
        .ClearFormatting
        .Font.Bold = False
    End With
    .Execute FindText:="", ReplaceWith:"", Format:=True, _
        Replace:=wdReplaceAll
End With
```

This example finds every instance of the word "Start" in the active document and replaces it with "End." The find operation ignores formatting but matches the case of the text to find ("Start").

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
With myRange.Find
    .ClearFormatting
    .Text = "Start"
    With .Replacement
        .ClearFormatting
        .Text = "End"
    End With
    .Execute Replace:=wdReplaceAll, _
        Format:=True, MatchCase:=True, _
        MatchWholeWord:=True
End With
```



ReplaceSelection Property

-
True if the result of typing or pasting replaces the selection. **False** if the result of typing or pasting is added before the selection, leaving the selection intact.
Read/write **Boolean**.

expression.**ReplaceSelection**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Microsoft Word to add the result of typing or pasting before the selection, leaving the selection intact.

```
Options.ReplaceSelection = False
```

This example returns the status of the **Typing replaces selection** option on the **Edit** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.ReplaceSelection
```



ReplaceText Property

True if Microsoft Word automatically replaces specified text with entries from the AutoCorrect list. Read/write **Boolean**.

expression.**ReplaceText**

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example sets Word to automatically replace specified text with entries from the AutoCorrect list as you type.

```
AutoCorrect.ReplaceText = True
```

This example toggles the value of the **ReplaceText** property.

```
AutoCorrect.ReplaceText = Not AutoCorrect.ReplaceText
```



ReplaceTextFromSpellingChecker Property

True if Microsoft Word automatically replaces misspelled text with suggestions from the spelling checker as the user types. Word only replaces words that contain a single misspelling and for which the spelling dictionary only lists one alternative. Read/write **Boolean**.

expression.**ReplaceTextFromSpellingChecker**

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example sets Word to automatically replace misspelled text with suggestions from the spelling checker.

```
AutoCorrect.ReplaceTextFromSpellingChecker = True
```



ReplyMessageSignature Property

Returns or sets the signature that Microsoft Word appends to e-mail message replies. Read/write **String**.

expression.**ReplyMessageSignature**

expression Required. An expression that returns an [EmailSignature](#) object.

Remarks

When setting this property, you must use the name of an e-mail signature that you have created in the **E-mail Options** dialog box, available from the **General** tab of the **Options** dialog box (**Tools** menu).

Example

This example changes the signature Word appends to e-mail message replies.

```
With Application.EmailOptions.EmailSignature  
    .ReplyMessageSignature = "Reply2"  
End With
```



ReplyStyle Property

Returns a [Style](#) object that represents the style used when replying to e-mail messages.

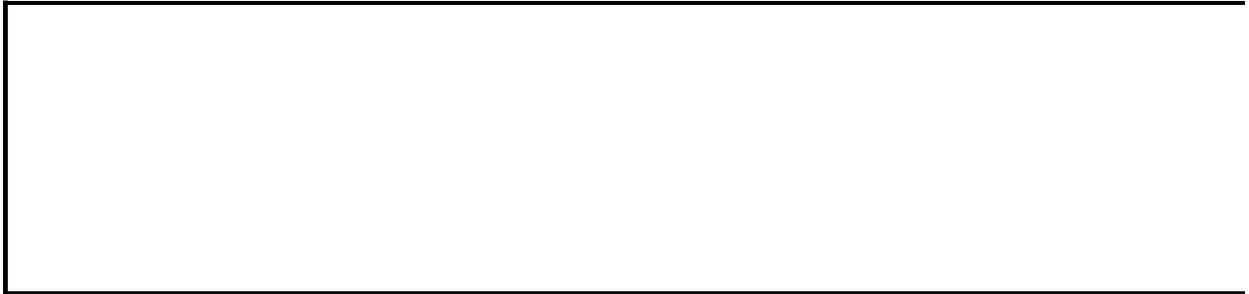
expression.**ReplyStyle**

expression Required. An expression that returns an [EmailOptions](#) object.

Example

This example displays the name of the default style used when replying to e-mail messages.

```
MsgBox Application.EmailOptions.ReplyStyle.NameLocal
```



ResetOnHigher Property

-

Sets or returns the list level that must appear before the specified list level restarts numbering at 1. **False** if the numbering continues sequentially each time the list level appears. Read/write **Long**.

expression.**ResetOnHigher**

expression Required. An expression that returns a [ListLevel](#) object.

Remarks

This feature allows lists to be interleaved, maintaining numeric sequence. You cannot set the **ResetOnHigher** property of a list level to a value greater than or equal to its index in the [ListLevels](#) collection.

Example

This example sets each of the nine list levels in the first outline-numbered list template to continue its sequential numbering whenever that level is used.

```
For Each li In _  
    ListGalleries(wdOutlineNumberGallery) _  
        .ListTemplates(1).ListLevels  
    li.ResetOnHigher = False  
Next li
```



↳ [Show All](#)

RestartMode Property

-

Returns or sets the way line numbering runs — that is, whether it starts over at the beginning of a new page or section or runs continuously. Read/write [WdNumberingRule](#).

WdNumberingRule can be one of these WdNumberingRule constants.

wdRestartContinuous

wdRestartPage

wdRestartSection

expression.**RestartMode**

expression Required. An expression that returns a [LineNumbering](#) object.

Remarks

You must be in print layout view to see line numbering.

Example

This example enables line numbering for the active document. The starting number is set to 1, every tenth line number is shown, and the numbering starts over at the beginning of each section.

```
set myDoc = ActiveDocument
With myDoc.PageSetup.LineNumbering
    .Active = True
    .StartingNumber = 1
    .CountBy = 10
    .RestartMode = wdRestartSection
End With
```



RestartNumberingAtSection Property

True if page numbering starts at 1 again at the beginning of the specified section. Read/write **Boolean**.

expression.**RestartNumberingAtSection**

expression Required. An expression that returns a [PageNumbers](#) collection object.

Remarks

If set to **False**, the **RestartNumberingAtSection** property will override the [StartingNumber](#) property so that page numbering can continue from the previous section.

Example

This example adds page numbers to the headers in the active document, and then it sets page numbering to start at 1 again at the beginning of each section.

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary) _  
    .PageNumbers.Add Pagenumberalignment:=wdAlignPageNumberCenter  
For Each s In ActiveDocument.Sections  
    With s.Headers(wdHeaderFooterPrimary).PageNumbers  
        .RestartNumberingAtSection = True  
        .StartingNumber = 1  
    End With  
Next s
```



↳ [Show All](#)

Result Property

▸ [Result property as it applies to the **Field** object.](#)

Returns a **Range** object that represents a field's result. You can access a field result without changing the view from field codes. Use the **Text** property to return text from a **Range** object. Read/write.

expression.**Result**

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

▸ [Result property as it applies to the **FormField** object.](#)

Returns a **String** that represents the result of the specified form field. Read/write.

expression.**Result**

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

Example

▶ [As it applies to the **Field** object.](#)

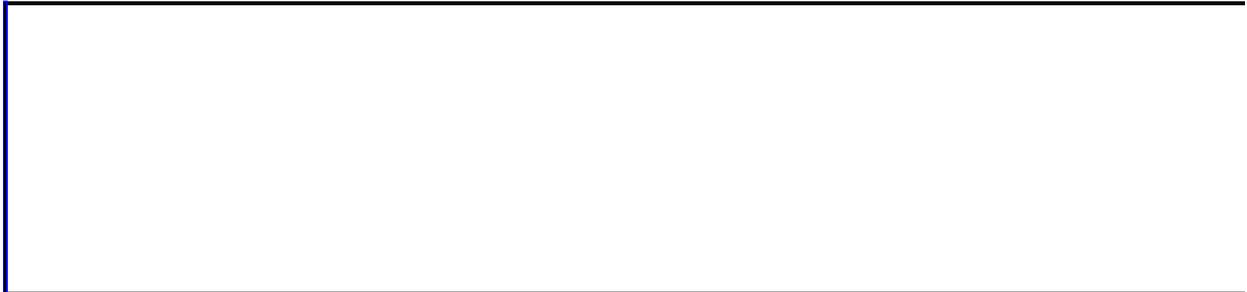
This example applies bold formatting to the first field in the selection.

```
If Selection.Fields.Count >= 1 Then
    Set myRange = Selection.Fields(1).Result
    myRange.Bold = True
End If
```

▶ [As it applies to the **FormField** object.](#)

This example displays the result of each form field in the active document.

```
For Each aField In ActiveDocument.FormFields
    MsgBox aField.Result
Next aField
```



↳ [Show All](#)

ReturnAddress Property

▸ [ReturnAddress property as it applies to the **Envelope** object.](#)

Returns a **Range** object that represents the envelope return address.

expression.**ReturnAddress**

expression Required. An expression that returns an **Envelope** object.

Remarks

An error occurs if you use this property before adding an envelope to the document.

▶ [ReturnAddress property as it applies to the LetterContent object.](#)

Returns or sets the return address for a letter created with the Letter Wizard.
Read/write **String**.

expression.**ReturnAddress**

expression Required. An expression that returns a [LetterContent](#) object.

Example

▶ [As it applies to the **Envelope** object.](#)

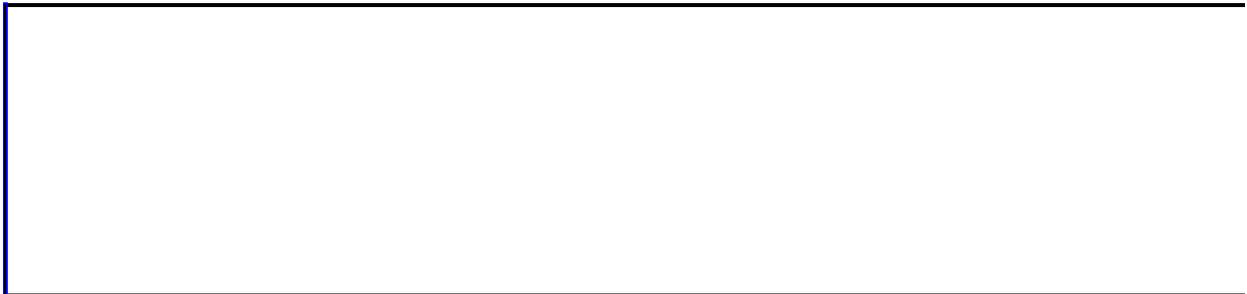
This example displays the return address if an envelope has been added to the active document; otherwise, a message box is displayed.

```
On Error GoTo errhandler
addr = ActiveDocument.Envelope.ReturnAddress.Text
MsgBox Prompt:=addr, Title:="Return Address"
errhandler:
If Err = 5852 Then MsgBox _
    "The active document doesn't include an envelope"
```

▶ [As it applies to the **LetterContent** object.](#)

This example creates a new **LetterContent** object, sets the return address and several other properties, and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Dim oLC as New LetterContent
With oLC
    .LetterStyle = wdFullBlock
    .Salutation = "Hello"
    .SalutationType = wdSalutationOther
    .ReturnAddress = Application.UserAddress
End With
Documents.Add.RunLetterWizard LetterContent:=oLC
```



ReturnAddressFromLeft Property

Returns or sets the distance (in points) between the left edge of the envelope and the return address. Read/write **Single**.

expression.**ReturnAddressFromLeft**

expression Required. An expression that returns an [Envelope](#) object.

Remarks

If you use this property before an envelope has been added to the document, an error occurs.

Example

This example creates a new document and adds an envelope with a predefined delivery address and return address. The example then sets the distance between the left edge of the envelope and the return address to 0.75 inch.

```
addr = "Karin Gallagher" & vbCr & "123 Skye St." _  
      & vbCr & "Our Town, WA 98004"  
retaddr = "Don Funk" & vbCr & "123 Main" _  
         & vbCr & "Other Town, WA 98040"  
With Documents.Add.Envelope  
    .Insert Address:=addr, ReturnAddress:=retaddr  
    .ReturnAddressFromLeft = InchesToPoints(0.75)  
End With  
ActiveDocument.ActiveWindow.View.Type = wdPrintView
```



ReturnAddressFromTop Property

Returns or sets the distance (in points) between the top edge of the envelope and the return address. Read/write **Single**.

expression.**ReturnAddressFromTop**

expression Required. An expression that returns an [Envelope](#) object.

Remarks

If you use this property before an envelope has been added to the document, an error occurs.

Example

This example creates a new document and adds an envelope with a predefined delivery address and return address. The example then sets the distance between the top edge of the envelope and the return address to 0.5 inch and sets the distance between the left edge of the envelope and the return address to 0.75 inch.

```
addr = "Eric Lang" & vbCr & "123 Main" _  
      & vbCr & "Seattle, WA 98040"  
retaddr = "Nate Sun" & vbCr & "123 Main" _  
         & vbCr & "Bellevue, WA 98004"  
With Documents.Add.Envelope  
    .Insert Address:=addr, ReturnAddress:=retaddr  
    .ReturnAddressFromTop = InchesToPoints(0.5)  
    .ReturnAddressFromLeft = InchesToPoints(0.75)  
End With
```



ReturnAddressShortForm Property

Returns or sets the short form address. Not used in the U.S. English version of Microsoft Word. Read/write **String**.

expression.**ReturnAddressShortForm**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

ReturnAddressStyle Property

Returns a [Style](#) object that represents the return address style for the envelope.

expression.**ReturnAddressStyle**

expression Required. An expression that returns an [Envelope](#) object.

Remarks

If an envelope is added to the document, text formatted with the Envelope Return style is automatically updated.

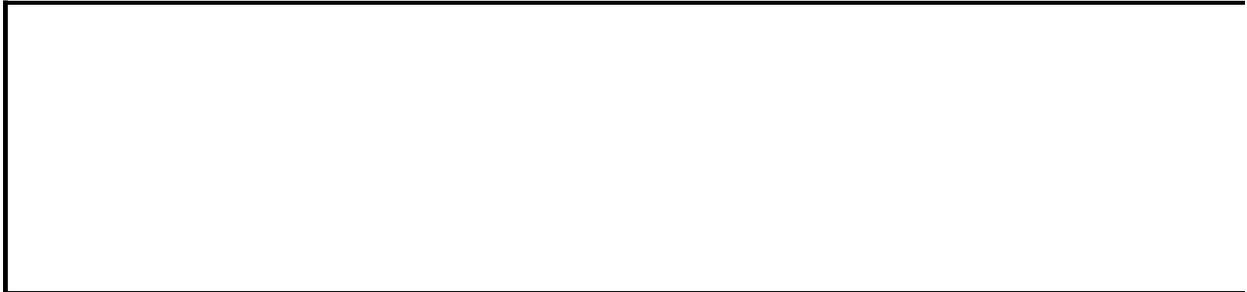
Example

This example displays the style name and description of the envelope return address.

```
Set myStyle = ActiveDocument.Envelope.ReturnAddressStyle  
MsgBox Prompt:=myStyle.Description, Title:=myStyle.NameLocal
```

This example sets the line spacing and space-after formatting for the envelope return address style.

```
With ActiveDocument.Envelope.ReturnAddressStyle.ParagraphFormat  
    .LineSpacingRule = wdLineSpaceExactly  
    .LineSpacing = 13  
    .SpaceAfter = 6  
End With
```



ReturnWhenDone Property

True if the document associated with the specified routing slip is sent back to the original sender when the routing is finished. Read/write **Boolean** before routing begins; read-only **Boolean** while routing is in progress.

expression.**ReturnWhenDone**

expression Required. An expression that returns a [RoutingSlip](#) object.

Example

This example sets the routing slip for Sales 1995.doc to return the document back to the original sender after the last recipient reviews it.

```
If Documents("Sales 1995.doc").HasRoutingSlip = True Then
    With Documents("Sales 1995.doc").RoutingSlip
        .Delivery = wdOneAfterAnother
        .ReturnWhenDone = True
    End With
End If
```



↳ [Show All](#)

Reverse Property

MsoTrue reverses the nodes in a diagram. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue Not used with this property.

msoFalse Leaves the diagram nodes as they are.

msoTriStateMixed Not used with this property.

msoTriStateToggle Not used with this property.

msoTrue Reverses the nodes in a diagram.

expression.**Reverse**

expression Required. An expression that returns a [Diagram](#) object.

Example

The following example creates a pyramid diagram and reverses the nodes so the node that was on the bottom of the pyramid is on the top and the node that was on the top is on the bottom.

```
Sub CreatePyramidDiagram()  
    Dim shpDiagram As Shape  
    Dim dgnNode As DiagramNode  
    Dim intCount As Integer  
  
    'Add pyramid diagram to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramPyramid, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add first child node to the diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes  
    For intCount = 1 To 3  
        dgnNode.AddNode  
    Next intCount  
  
    With dgnNode.Diagram  
        'Enable automatic formatting  
        .AutoFormat = msoTrue  
  
        'Reverse the order of the nodes  
        .Reverse = msoTrue  
    End With  
End Sub
```



Reviewers Property

Returns a [Reviewers](#) object that represents all reviewers.

expression.**Reviewers**

expression Required. An expression that returns a [View](#) object.

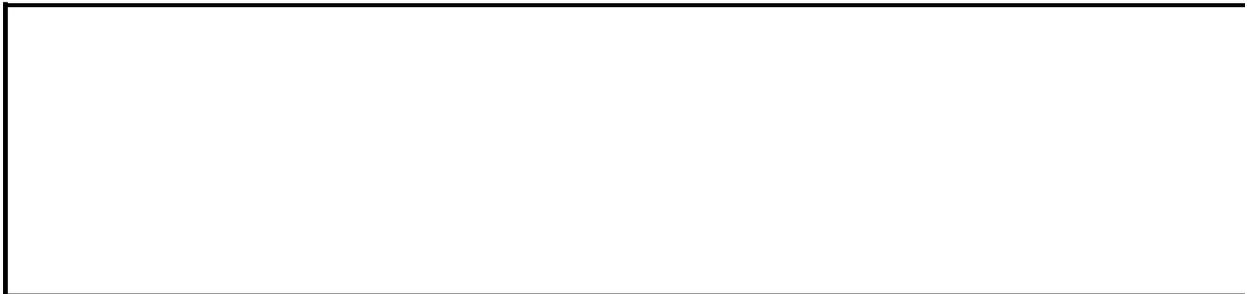
Remarks

The **Reviewers** object is a global list of all reviewers, regardless of whether the reviewer reviewed the document displayed in the specified window.

Example

This example hides all revisions and comments in the document and displays only revisions and comments made by "Jeff Smith."

```
Sub HideRevisions()  
    Dim revName As Reviewer  
    With ActiveWindow.View  
        .ShowRevisionsAndComments = False  
        .ShowFormatChanges = True  
        .ShowInsertionsAndDeletions = True  
  
        For Each revName In .Reviewers  
            revName.Visible = True  
        Next  
  
        .Reviewers.Item("Jeff Smith").Visible = True  
    End With  
End Sub
```



↳ [Show All](#)

RevisedLinesColor Property

-

Returns or sets the color of changed lines in a document with tracked changes.
Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**RevisedLinesColor**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets the color of changed lines to pink.

```
Options.RevisedLinesColor = wdPink
```

This example returns the current status of the **Color** option under **Changed lines** on the **Track Changes** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.RevisedLinesColor
```



↳ [Show All](#)

RevisedLinesMark Property

Returns or sets the placement of changed lines in a document with tracked changes. Read/write [WdRevisedLinesMark](#).

WdRevisedLinesMark can be one of these WdRevisedLinesMark constants.

wdRevisedLinesMarkLeftBorder

wdRevisedLinesMarkNone

wdRevisedLinesMarkOutsideBorder

wdRevisedLinesMarkRightBorder

expression.**RevisedLinesMark**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets changed lines to appear in the left margin of every page.

```
Options.RevisedLinesMark = wdRevisedLinesMarkLeftBorder
```

This example returns the current status of the **Mark** option under **Changed lines** on the **Track Changes** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.RevisedLinesMark
```



↳ [Show All](#)

RevisedPropertiesColor Property

Returns or sets the color used to mark formatting changes while change tracking is enabled. Read/write [WdColorIndex](#).

WdColorIndex can be one of these WdColorIndex constants.

wdAuto

wdBlack

wdBlue

wdBrightGreen

wdByAuthor

wdDarkBlue

wdDarkRed

wdDarkYellow

wdGray25

wdGray50

wdGreen

wdNoHighlight

wdPink

wdRed

wdTeal

wdTurquoise

wdViolet

wdWhite

wdYellow

expression.**RevisedPropertiesColor**

expression Required. An expression that returns an [Options](#) object.

Remarks

If deleted or inserted text has formatting changes, the **RevisedPropertiesColor** property is overridden by the [DeletedTextColor](#) or [InsertedTextColor](#) property.

Example

This example tracks changes in the active document, sets the color of text with changed formatting to teal, and applies bold formatting to the selection.

```
ActiveDocument.TrackRevisions = True  
Options.RevisedPropertiesColor = wdTeal  
Selection.Font.Bold = True
```

This example returns the option selected in the **Color** box under **Track Changes options** on the **Track Changes** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.RevisedPropertiesColor
```



↳ [Show All](#)

RevisedPropertiesMark Property

Returns or sets the mark used to show formatting changes while change tracking is enabled. Read/write [WdRevisedPropertiesMark](#).

WdRevisedPropertiesMark can be one of these WdRevisedPropertiesMark constants.

wdRevisedPropertiesMarkBold

wdRevisedPropertiesMarkColorOnly

wdRevisedPropertiesMarkDoubleUnderline

wdRevisedPropertiesMarkItalic

wdRevisedPropertiesMarkNone

wdRevisedPropertiesMarkStrikeThrough

wdRevisedPropertiesMarkUnderline

expression.**RevisedPropertiesMark**

expression Required. An expression that returns an [Options](#) object.

Example

This example causes text with changed formatting to be double-underlined when change tracking is enabled.

```
Options.RevisedPropertiesMark = _  
    wdRevisedPropertiesMarkDoubleUnderline
```

This example returns the option selected in the **Formatting** box under **Track Changes options** on the **Track Changes** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.RevisedPropertiesMark
```



Revisions Property

-

Returns a [Revisions](#) collection that represents the tracked changes in the document or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the number of tracked changes in the first section in the active document.

```
MsgBox ActiveDocument.Sections(1).Range.Revisions.Count
```

This example accepts all tracked changes in the first paragraph in the selection.

```
Set myRange = Selection.Paragraphs(1).Range  
myRange.Revisions.AcceptAll
```



↳ [Show All](#)

RevisionsBalloonPrintOrientation Property

Returns or sets a [WdRevisionsBalloonPrintOrientation](#) constant that represents the direction of revision and comment balloons when they are printed. Read/write.

WdRevisionsBalloonPrintOrientation can be one of these WdRevisionsBalloonPrintOrientation constants.

wdBalloonPrintOrientationAuto Microsoft Word automatically selects the orientation that keeps the zoom factor closest to 100%.

wdBalloonPrintOrientationForceLandscape Word forces all sections to be printed in Landscape mode, regardless of original orientation, and prints the revision and comment balloons on the side opposite to the document text.

wdBalloonPrintOrientationPerserve Word preserves the orientation of the original, uncommented document.

expression.**RevisionsBalloonPrintOrientation**

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

Example

This example prints documents with comments in Landscape format with the revision and comment balloons on one side of the page and the document text on the other.

```
Sub PrintLandscapeCommentBalloons()  
    Options.RevisionsBalloonPrintOrientation = _  
        wdBalloonPrintOrientationForceLandscape  
End Sub
```



RevisionsBalloonShowConnectingLines Property

True for Microsoft Word to display connecting lines from the text to the revision and comment balloons. Read/write **Boolean**.

expression.**RevisionsBalloonShowConnectingLines**

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

Example

This example hides the lines connecting the document text with the corresponding revision or comment balloons.

```
Sub ShowConnectingLines()  
    ActiveWindow.View _  
        .RevisionsBalloonShowConnectingLines = False  
End Sub
```



↳ [Show All](#)

RevisionsBalloonSide Property

-

Sets or returns a [WdRevisionsBalloonMargin](#) constant representing the global setting in Microsoft Word that specifies whether Word displays revision balloons in the left or right margin. Read/write.

WdRevisionsBalloonMargin can be one of these WdRevisionsBalloonMargin constants.

wdLeftMargin

wdRightMargin

expression.**RevisionsBalloonSide**

expression Required. An expression that returns a [View](#) object.

Example

This example toggles the revision balloons between the left and right side. This example assumes that the document in the active window contains revisions made by one or more reviewers and that revisions are displayed in balloons.

```
Sub ToggleRevisionBalloons()  
  With ActiveWindow.View  
    If .RevisionsBalloonSide = wdLeftMargin Then  
      .RevisionsBalloonSide = wdRightMargin  
    Else  
      .RevisionsBalloonSide = wdLeftMargin  
    End If  
  End With  
End Sub
```



RevisionsBalloonWidth Property

-
Sets or returns a **Single** representing the global setting in Microsoft Word that specifies the width of the revision balloons. Read/write.

expression.**RevisionsBalloonWidth**

expression Required. An expression that returns one a [View](#) object.

Remarks

The width of revision balloons includes padding of one-half inch between the document margin and the edge of the balloon and one-eighth of an inch between the edge of the balloon and the edge of the paper. Microsoft Word adds space along the left or right edge of the paper. This width is extended into the margin and does not change the width of the document or paper size. Use the [RevisionsBalloonWidthType](#) property to specify the measurement to use when setting the **RevisionsBalloonWidth** property.

Example

This example sets the width of the revision balloons to one inch and displays the revision balloons in the left margin. This example assumes that the document in the active window contains revisions made by one or more reviewers and that revisions are displayed in balloons.

```
Sub BalloonWidth()  
  With ActiveWindow.View  
    .RevisionsBalloonWidthType = wdBalloonWidthPoints  
    .RevisionsBalloonWidth = InchesToPoints(1)  
    .RevisionsBalloonSide = wdLeftMargin  
  End With  
End Sub
```



↳ [Show All](#)

RevisionsBalloonWidthType Property

Sets or returns a [WdRevisionsBalloonWidthType](#) constant representing the global setting that specifies how Microsoft Word measures the width of revision balloons. Read/write.

WdRevisionsBalloonWidthType can be one of these WdRevisionsBalloonWidthType constants.

wdBalloonWidthPercent Measured as a percentage of the width of the document.

wdBalloonWidthPoints Measured in points.

expression.**RevisionsBalloonWidthType**

expression Required. An expression that returns a [View](#) object.

Remarks

The **RevisionsBalloonWidthType** property sets the measurement unit to use when setting the [RevisionsBalloonWidth](#) property.

Example

This example sets the width of the revision balloons to twenty-five percent of the document's width. This example assumes that the document in the active window contains revisions made by one or more reviewers and that revisions are displayed in balloons.

```
Sub BalloonWidthType()  
  With ActiveWindow.View  
    .RevisionsBalloonWidthType = wdBalloonWidthPercent  
    .RevisionsBalloonWidth = 25  
  End With  
End Sub
```



↳ [Show All](#)

RevisionsMode Property

Sets or returns a [WdRevisionsMode](#) constant representing the global option that specifies whether Microsoft Word displays balloons in the margin or inline with the document's text. Read/write.

WdRevisionsMode can be one of these WdRevisionsMode constants.

wdBalloonRevisions Displays revisions in balloons in the left or right margin.

wdInLineRevisions Displays revisions within the text using strikethrough for deletions and underlining for insertions. This is the default setting for prior versions of Word.

expression.**RevisionsMode**

expression Required. An expression that returns a [View](#) object.

Example

This example toggles between displaying the revisions in balloons in the margins and displaying them inline with the text. This example assumes that the document in the active window contains revisions made by one or more reviewers and that revisions are displayed in balloons.

```
Sub TogglesRevisionMode()  
  With ActiveWindow.View  
    If .RevisionsMode = wdInLineRevisions Then  
      .RevisionsMode = wdBalloonRevisions  
    Else  
      .RevisionsMode = wdInLineRevisions  
    End If  
  End With  
End Sub
```



↳ [Show All](#)

RevisionsView Property

Sets or returns a [WdRevisionsView](#) constant representing the global option that specifies whether Word displays the original version of a document or a version with revisions and formatting changes applied. Read/write.

WdRevisionsView can be one of these WdRevisionsView constants.

wdRevisionsViewFinal Displays the document with formatting and content changes applied.

wdRevisionsViewOriginal Displays the document before changes were made.

expression.**RevisionsView**

expression Required. An expression that returns a [View](#) object.

Example

This example toggles between displaying the original and a final version of the document. This example assumes that the document in the active window contains revisions made by one or more reviewers and that revisions are displayed in balloons.

```
Sub ToggleRevView()  
  With ActiveWindow.View  
    If .RevisionsMode = wdBalloonRevisions Then  
      If .RevisionsView = wdRevisionsViewFinal Then  
        .RevisionsView = wdRevisionsViewOriginal  
      Else  
        .RevisionsView = wdRevisionsViewFinal  
      End If  
    End If  
  End With  
End Sub
```



↳ [Show All](#)

RGB Property

-

Returns or sets the [red-green-blue \(RGB\) value](#) of the specified color. Read/write **Long**.

Example

This example sets the color of the second shape in the active document to gray.

```
ActiveDocument.Shapes(2).Fill.ForeColor.RGB = RGB(128, 128, 128)
```

This example sets the color of the shadow for Rectangle 1 in the active document to blue.

```
ActiveDocument.Shapes("Rectangle 1").Shadow.ForeColor.RGB = _  
    RGB(0, 0, 255)
```

This example returns the value of the foreground color of the first shape in the active document.

```
MsgBox ActiveDocument.Shapes(1).Fill.ForeColor.RGB
```



RichText Property

-
True if formatting is stored with the AutoCorrect entry replacement text. Read-only **Boolean**.

expression.**RichText**

expression Required. An expression that returns an [AutoCorrectEntry](#) object.

Example

This example determines whether AutoCorrect entry one is formatted.

```
MsgBox AutoCorrect.Entries(1).RichText
```



RightAlignPageNumbers Property

True if page numbers are aligned with the right margin in an index, table of contents, or table of figures. Read/write **Boolean**.

expression.**RightAlignPageNumbers**

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

Example

This example right-aligns page numbers for the first table of contents in the active document.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .IncludePageNumbers = True
        .RightAlignPageNumbers = True
    End With
End If
```



RightIndent Property

Returns or sets the right indent (in points) for the specified paragraphs.
Read/write **Single**.

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 1 inch from the right margin. The [InchesToPoints](#) method is used to convert inches to points.

```
ActiveDocument.Paragraphs.RightIndent = InchesToPoints(1)
```



RightMargin Property

Returns or sets the distance (in points) between the right edge of the page and the right boundary of the body text. Read/write **Single**.

expression.**RightMargin**

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

Remarks

If the [MirrorMargins](#) property is set to **True**, the **RightMargin** property controls the setting for outside margins and the [LeftMargin](#) property controls the setting for inside margins.

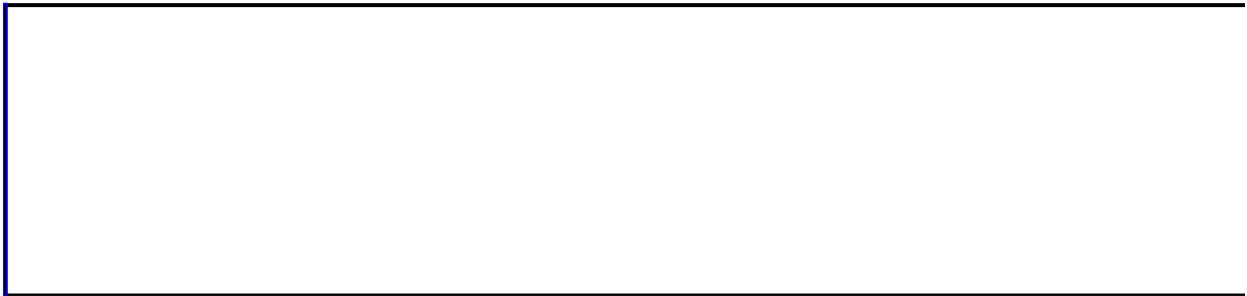
Example

This example displays the right margin setting for the active document. The [PointsToInches](#) method is used to convert the result to inches.

```
With ActiveDocument.PageSetup
    MsgBox "The right margin is set to " & _
        & PointsToInches(.RightMargin) & " inches."
End With
```

This example sets the right margin for section two in the selection. The [InchesToPoints](#) method is used to convert inches to points.

```
Selection.Sections(2).PageSetup.RightMargin = InchesToPoints(1)
```



RightPadding Property

Returns or sets the amount of space (in points) to add to the right of the contents of a single cell or all the cells in a table. Read/write **Single**.

expression.**RightPadding**

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

Remarks

The setting of the **RightPadding** property for a single cell overrides the setting of the **RightPadding** property for the entire table.

Example

This example sets the right padding for the first table in the active document to 40 pixels.

```
ActiveDocument.Tables(1).RightPadding = _  
    PixelsToPoints(40, False)
```



Root Property

Returns a [DiagramNode](#) object that represents the root diagram node to which the source diagram node belongs. Read-only.

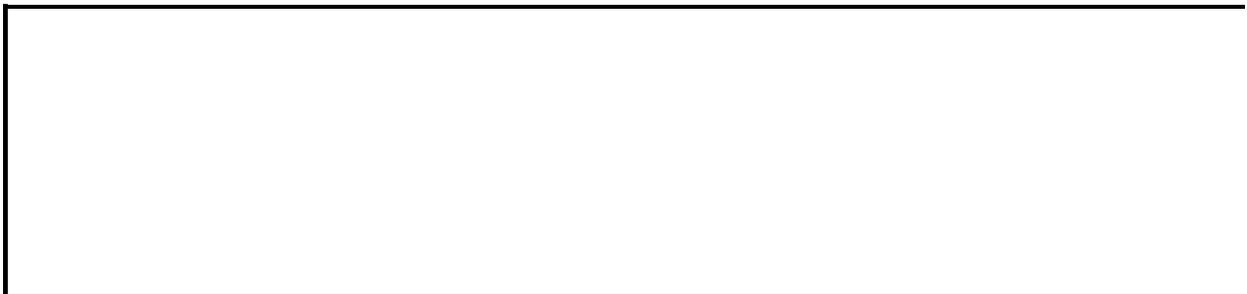
expression.**Root**

expression Required. An expression that returns a [DiagramNode](#) object.

Example

The following example creates an organization chart and adds child nodes to the root diagram node.

```
Sub Root()  
    Dim shpDiagram As Shape  
    Dim dgnRoot As DiagramNode  
    Dim intCount As Integer  
  
    'Add organization chart to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramOrgChart, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add the first node to the diagram  
    shpDiagram.DiagramNode.Children.AddNode  
  
    'Assign the root diagram node to a variable  
    Set dgnRoot = shpDiagram.DiagramNode.Root  
  
    'Add three child nodes to the root node  
    For intCount = 1 To 3  
        dgnRoot.Children.AddNode  
    Next intCount  
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 a [TextEffectFormat](#) object.

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 myDocument and rotates the characters 90 degrees counterclockwise.

```
Set myDocument = ActiveDocument
Set newWordArt = _
    myDocument.Shapes.AddTextEffect( _
        PresetTextEffect:=msoTextEffect1, _
        Text:="Test", _
        FontName:="Arial Black", FontSize:=36, _
        FontBold:=False, FontItalic:=False, Left:=10, Top:=10)
newWordArt.TextEffect.RotatedChars = True
```



Rotation Property

-

Returns or sets 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 **Single**.

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 myDocument to the rotation of shape one.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
    sh1Rotation = .Item(1).Rotation
    For o = 1 To .Count
        .Item(o).Rotation = sh1Rotation
    Next
End With
```



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 a [ThreeDFormat](#) object.

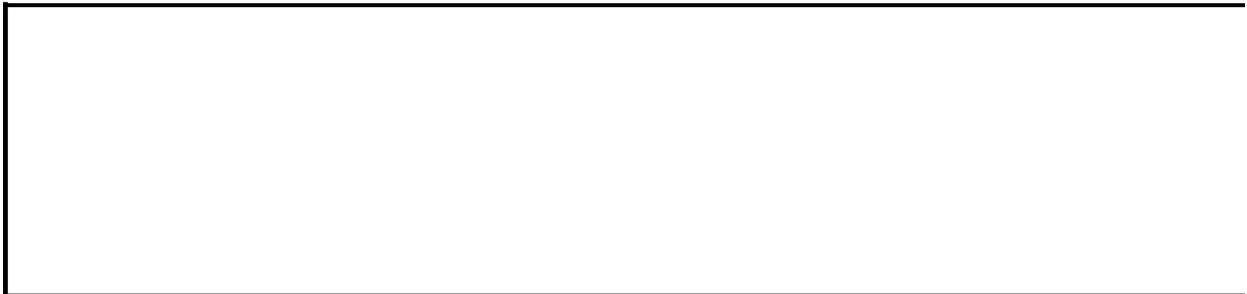
Remarks

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.

```
With ActiveDocument.Shapes
  With .AddShape(msoShapeOval, 30, 60, 50, 25).ThreeD
    .Visible = True
    .RotationX = -30
  End With
  With .AddShape(msoShapeOval, 90, 60, 50, 25).ThreeD
    .Visible = True
    .RotationX = 0
  End With
  With .AddShape(msoShapeOval, 150, 60, 50, 25).ThreeD
    .Visible = True
    .RotationX = 30
  End With
End With
```



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 a [ThreeDFormat](#) object.

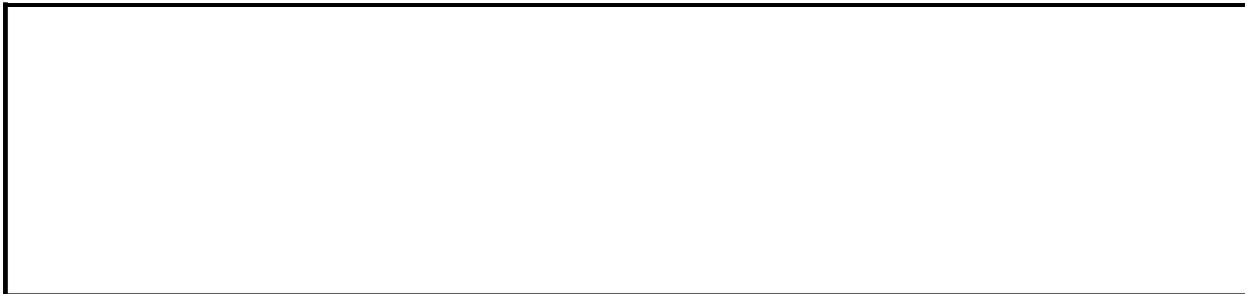
Remarks

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 myDocument and sets their rotation around the y-axis to -30 , 0 , and 30 degrees, respectively.

```
Set myDocument = ActiveDocument
With myDocument.Shapes
  With .AddShape(msoShapeOval, 30, 30, 50, 25).ThreeD
    .Visible = True
    .RotationY = -30
  End With
  With .AddShape(msoShapeOval, 30, 70, 50, 25).ThreeD
    .Visible = True
    .RotationY = 0
  End With
  With .AddShape(msoShapeOval, 30, 110, 50, 25).ThreeD
    .Visible = True
    .RotationY = 30
  End With
End With
```



Routed Property

-

True if the specified document has been routed to the next recipient. **False** if the document has yet to be routed (for example, if the document has no routing slip, or if a routing slip was just created). Read-only **Boolean**.

Example

This example routes the active document if it hasn't yet been routed.

```
If ActiveDocument.Routed = False Then ActiveDocument.Route
```



RoutingSlip Property

-

Returns a [RoutingSlip](#) object that represents the routing slip information for the specified document. A routing slip is used to send a document through an electronic mail system. Read-only.

Example

This example adds a routing slip to Status.doc and then routes the document to the specified recipients.

```
Documents("Status.doc").HasRoutingSlip = True
With Documents("Status.doc").RoutingSlip
    .Subject = "Status Doc "
    .AddRecipient Recipient:="Don Funk"
    .AddRecipient Recipient:="Frida Ebbeson"
    .Delivery = wdAllAtOnce
End With
Documents("Status.doc").Route
```



Row Property

Returns a [Row](#) object that represents the row containing the specified cell.

expression.**Row**

expression Required. An expression that returns a [Cell](#) object.

Example

This example applies shading to the table row that contains the insertion point.

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells(1).Row.Shading.Texture = wdTexture10Percent
Else
    MsgBox "The insertion point is not in a table."
End If
```



RowIndex Property

Returns the number of the row that contains the specified cell. Read-only **Long**.

expression.**RowIndex**

expression Required. An expression that returns a [Cell](#) object.

Example

This example creates a 3x3 table in a new document, selects each cell in the first column, and displays the row number that contains each selected cell.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Range:=Selection.Range, _
    NumRows:=3, NumColumns:=3)
For Each aCell In myTable.Columns(1).Cells
    aCell.Select
    MsgBox "This is row " & aCell.RowIndex
Next aCell
```

This example displays the row number of the first row in the selection.

```
If Selection.Information(wdWithInTable) = True Then
    MsgBox Selection.Cells(1).RowIndex
End If
```



Rows Property

Returns a [Rows](#) collection that represents all the table rows in a range, selection, or table. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example deletes the second row from the first table in the active document.

```
ActiveDocument.Tables(1).Rows(2).Delete
```

This example places a border around the cells in the row that contains the insertion point.

```
Selection.Collapse Direction:=wdCollapseStart
If Selection.Information(wdWithInTable) = True Then
    Selection.Rows(1).Borders.OutsideLineStyle = wdLineStyleSingle
Else
    MsgBox "The insertion point is not in a table."
End If
```



RowStripe Property

Returns or sets a **Long** that represents the number of rows to include in the banding when a style specifies odd- or even-row banding. Read/write.

expression.**RowStripe**

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

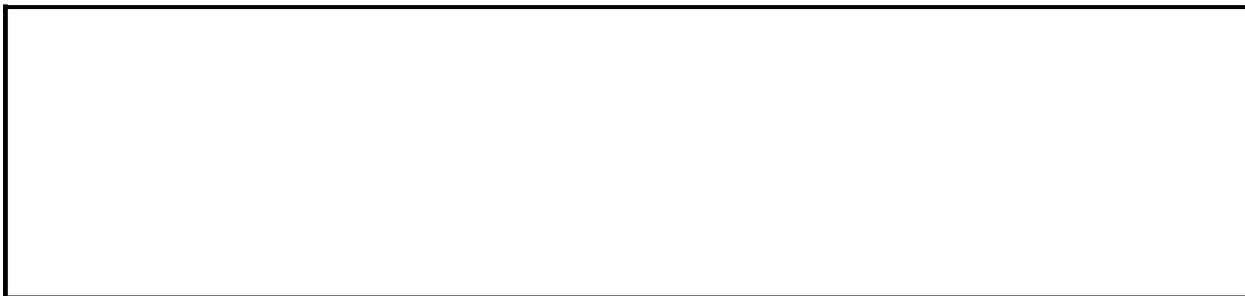
Remarks

Use the [Condition](#) method to set odd- or even-column banding for a table style.

Example

This example creates and formats a new table style then applies the new style to a new table. The resulting style causes three columns every third column and two rows every second row to have 20% shading.

```
Sub NewTableStyle()  
    Dim styTable As Style  
  
    With ActiveDocument  
        Set styTable = .Styles.Add(Name:="TableStyle 1", _  
            Type:=wdStyleTypeTable)  
  
        With .Styles("TableStyle 1").Table  
            .Condition(wdEvenColumnBanding).Shading _  
                .Texture = wdTexture20Percent  
            .ColumnStripe = 3  
            .Condition(wdEvenRowBanding).Shading _  
                .Texture = wdTexture20Percent  
            .RowStripe = 2  
        End With  
  
        With .Tables.Add(Range:=Selection.Range, NumRows:=15, _  
            NumColumns:=15)  
            .Style = ActiveDocument.Styles("TableStyle 1")  
        End With  
    End With  
End Sub
```



Salutation Property

Returns or sets the salutation text for a letter created by the Letter Wizard.
Read/write **String**.

expression.**Salutation**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, sets several properties (including the salutation text), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent  
myContent.Salutation = "Hello, "  
Documents.Add.RunLetterWizard LetterContent:=myContent
```



↳ [Show All](#)

SalutationType Property

Returns or sets the type of salutation for a letter created by the Letter Wizard.
Read/write [WdSalutationType](#).

WdSalutationType can be one of these WdSalutationType constants.

wdSalutationBusiness

wdSalutationFormal

wdSalutationInformal

wdSalutationOther

expression.**SalutationType**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new **LetterContent** object, sets several properties (including the salutation text), and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
myContent.SalutationType = wdSalutationBusiness
Documents.Add.RunLetterWizard _
    LetterContent:=myContent, WizardMode:=True
```



Saved Property

-

True if the specified document or template hasn't changed since it was last saved. **False** if Microsoft Word displays a prompt to save changes when the document is closed. Read/write **Boolean**.

Example

This example saves the active document if it contains previously unsaved changes.

```
If ActiveDocument.Saved = False Then ActiveDocument.Save
```

This example changes the status of the Normal template to unchanged. If changes were made to the Normal template, the changes aren't saved when you quit Word.

```
NormalTemplate.Saved = True  
Application.Quit
```



SavedBy Property

Returns the name of the user who saved the specified version of the document.
Read-only **String**.

expression.**SavedBy**

expression Required. An expression that returns a [Version](#) object.

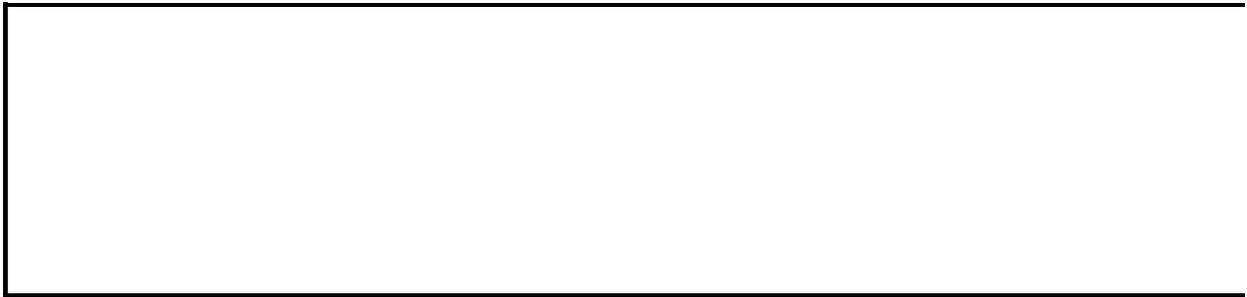
Example

This example displays the name of the user who saved the first version of the active document.

```
If ActiveDocument.Versions.Count >= 1 Then
    MsgBox ActiveDocument.Versions(1).SavedBy
End If
```

This example saves a version of the document with a comment and then displays the user name.

```
ActiveDocument.Versions.Save Comment:="Added client information"
last = ActiveDocument.Versions.Count
MsgBox ActiveDocument.Versions(last).SavedBy
```



SaveEncoding Property

Returns or sets the encoding to use when saving a document. Read/write [MsoEncoding](#).

MsoEncoding can be one of these MsoEncoding constants; however, you cannot use any of the constants that have the suffix **AutoDetect**. These constants are used by the [ReloadAs](#) method.

msoEncodingOEMMultilingualLatinI

msoEncodingOEMNordic

msoEncodingOEMTurkish

msoEncodingSimplifiedChineseAutoDetect

msoEncodingT61

msoEncodingTaiwanEten

msoEncodingTaiwanTCA

msoEncodingTaiwanWang

msoEncodingTraditionalChineseAutoDetect

msoEncodingTurkish

msoEncodingUnicodeLittleEndian

msoEncodingUTF7

msoEncodingVietnamese

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDICKoreanExtendedAndKorean

msoEncodingEBCDICMultilingualROECELatin2

msoEncodingEBCDICSerbianBulgarian

msoEncodingEBCDICThai

msoEncodingEBCDICTurkishLatin5

msoEncodingEBCDICUSCanada

msoEncodingEBCDICUSCanadaAndTraditionalChinese

msoEncodingOEMModernGreek
msoEncodingOEMMultilingualLatinII
msoEncodingOEMPortuguese
msoEncodingOEMUnitedStates
msoEncodingSimplifiedChineseGBK
msoEncodingTaiwanCNS
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTeleText
msoEncodingThai
msoEncodingTraditionalChineseBig5
msoEncodingUnicodeBigEndian
msoEncodingUSASCII
msoEncodingUTF8
msoEncodingWestern
msoEncodingArabic
msoEncodingArabicASMO
msoEncodingArabicAutoDetect
msoEncodingArabicTransparentASMO
msoEncodingAutoDetect
msoEncodingBaltic
msoEncodingCentralEuropean
msoEncodingCyrillic
msoEncodingCyrillicAutoDetect
msoEncodingEBCDICArabic
msoEncodingEBCDICDenmarkNorway
msoEncodingEBCDICFinlandSweden
msoEncodingEBCDICFrance
msoEncodingEBCDICGermany
msoEncodingEBCDICGreek
msoEncodingEBCDICGreekModern
msoEncodingEBCDICHebrew
msoEncodingEBCDICIcelandic
msoEncodingEBCDICInternational

msoEncodingEBCDICItaly
msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese
msoEncodingEBCDIKKoreanExtended
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICRussian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICTurkish
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
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

expression.**SaveEncoding**

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

Example

This example specifies Western encoding for saving the current document.

```
ActiveDocument.SaveEncoding = msoEncodingWestern
```



↳ [Show All](#)

SaveFormat Property

Returns the file format of the specified document or file converter. Will be a unique number that specifies an external file converter or a [WdSaveFormat](#) constant. Read-only **Long**.

WdSaveFormat can be one of the following WdSaveFormat constants.

wdFormatDocument

wdFormatDOSText

wdFormatDOSTextLineBreaks

wdFormatEncodedText

wdFormatHTML

wdFormatRTF

wdFormatTemplate

wdFormatText

wdFormatTextLineBreaks

wdFormatUnicodeText

expression.**SaveFormat**

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

Remarks

Use the value of the **SaveFormat** property for the *FileFormat* argument of the [SaveAs](#) method to save a document in a file format for which there isn't a corresponding **WdSaveFormat** constant.

Example

If the active document is a Rich Text Format (RTF) document, this example saves it as a Microsoft Word document.

```
If ActiveDocument.SaveFormat = wdFormatRTF Then
    ActiveDocument.SaveAs FileFormat:=wdFormatDocument
End If
```

This example creates a new document and lists in a table the converters that can be used to save documents and their corresponding **SaveFormat** values.

```
Sub FileConverterList()
    Dim cnvFile As FileConverter
    Dim docNew As Document

    'Create a new document and set a tab stop
    Set docNew = Documents.Add
    docNew.Paragraphs.Format.TabStops.Add _
        Position:=InchesToPoints(3)

    'List all the converters in the FileConverters collection
    With docNew.Content
        .InsertAfter "Name" & vbTab & "Number"
        .InsertParagraphAfter
        For Each cnvFile In FileConverters
            If cnvFile.CanSave = True Then
                .InsertAfter cnvFile.FormatName & vbTab & _
                    cnvFile.SaveFormat
                .InsertParagraphAfter
            End If
        Next
        .ConvertToTable
    End With

End Sub
```

This example saves the active document in the WordPerfect 5.1 or 5.2 secondary file format.

```
ActiveDocument.SaveAs _
    FileFormat:=FileConverters("WrdPrfctDat").SaveFormat
```



SaveFormsData Property

-
True if Microsoft Word saves the data entered in a form as a tab-delimited record for use in a database. Read/write **Boolean**.

Example

This example sets Word to save only the data entered in a form

```
ActiveDocument.SaveFormsData = True
```

This example returns the current status of the **Save data only for forms** check box in the **Save options** area on the **Save** tab in the **Options** dialog box.

```
temp = ActiveDocument.SaveFormsData
```



SaveInterval Property

Returns or sets the time interval in minutes for saving AutoRecover information.
Read/write **Long**.

expression.**SaveInterval**

expression Required. An expression that returns an [Options](#) object.

Remarks

Set the **SaveInterval** property to 0 (zero) to turn off saving AutoRecover information.

Example

This example sets Word to save AutoRecover information for all open documents every five minutes.

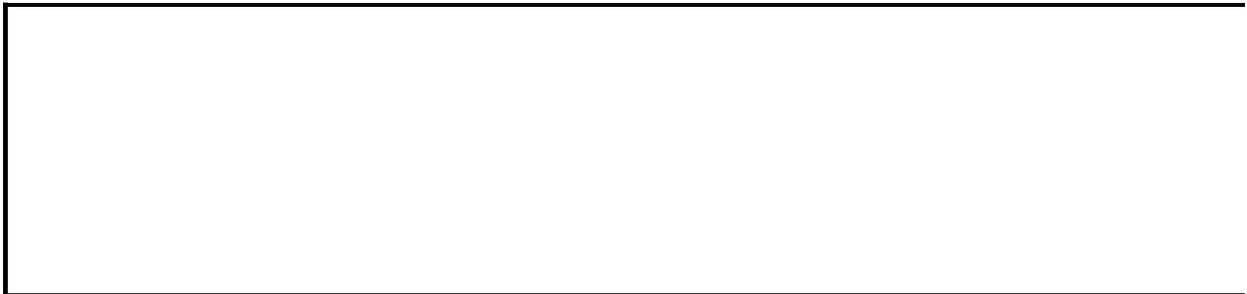
```
Options.SaveInterval = 5
```

This example prevents Word from saving AutoRecover information.

```
Options.SaveInterval = 0
```

This example returns the current status of the **Save AutoRecover info every** option on the **Save** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SaveInterval
```



SaveNewWebPagesAsWebArchives Property

True for Microsoft Word to save new Web pages using the Web Archive format.
Read/write **Boolean**.

expression.**SaveNewWebPagesAsWebArchives**

expression Required. An expression that returns a [DefaultWebOptions](#) object.

Remarks

Setting the **SaveNewWebPagesAsWebArchives** property won't change the format of any saved Web pages. To change their format, you must individually open them and then use the [SaveAs](#) method to set the Web page format.

Example

This example enables the **SaveNewWebPagesAsWebArchives** property so that when Web pages are saved, they are saved using the Web Archive format.

```
Sub SetWebOption()  
    Application.DefaultWebOptions _  
        .SaveNewWebPagesAsWebArchives = True  
End Sub
```



SaveNormalPrompt Property

-

True if Microsoft Word prompts the user for confirmation to save changes to the Normal template before it quits. **False** if Word automatically saves changes to the Normal template before it quits. Read/write **Boolean**.

expression.**SaveNormalPrompt**

expression Required. An expression that returns an [Options](#) object.

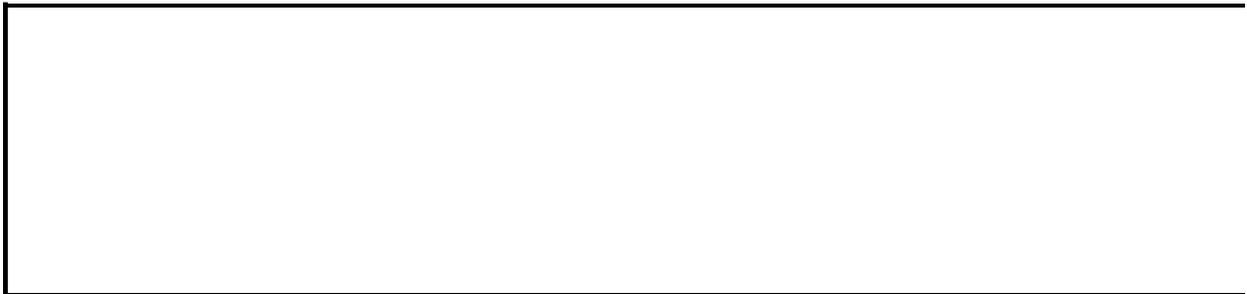
Example

This example sets Word to save the Normal template automatically before quitting, and then it quits.

```
Options.SaveNormalPrompt = False  
Application.Quit
```

This example returns the current status of the **Prompt to save Normal template** option on the **Save** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SaveNormalPrompt
```



SavePictureWithDocument Property

True if the specified picture is saved with the document. Read/write **Boolean**.

expression.**SavePictureWithDocument**

expression Required. An expression that returns a [LinkFormat](#) object.

Remarks

This property works only with shapes and inline shapes that are linked pictures.

Example

This example saves the linked picture that's defined as the first inline shape in the active document when the document is saved.

```
Set myPic = ActiveDocument.InlineShapes(1)
If myPic.Type = wdInlineShapeLinkedPicture Then
    myPic.LinkFormat.SavePictureWithDocument = True
End If
```



SavePropertiesPrompt Property

True if Microsoft Word prompts for document property information when saving a new document. Read/write **Boolean**.

expression.**SavePropertiesPrompt**

expression Required. An expression that returns an [Options](#) object.

Example

This example causes Word to prompt for document property information when saving a new document.

```
Options.SavePropertiesPrompt = True
```

This example returns the current status of the **Prompt for document properties** option on the **Save** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SavePropertiesPrompt
```



SaveSubsetFonts Property

-
True if Microsoft Word saves a subset of the embedded TrueType fonts with the document. Read/write **Boolean**.

Remarks

If fewer than 32 characters of a TrueType font are used in a document, Word embeds the subset (only the characters used) in the document. If more than 32 characters are used, Word embeds the entire font.

Example

This example sets a document named "MyDoc" to save only a subset of its embedded TrueType fonts (when just a few characters are used), and then it saves "MyDoc."

```
With Documents("MyDoc")  
    .EmbedTrueTypeFonts = True  
    .SaveSubsetFonts = True  
    .Save  
End With
```



ScaleHeight Property

-
Scales the height of the specified inline shape relative to its original size.
Read/write **Single**.

expression.**ScaleHeight**

expression Required. An expression that returns an [InlineShape](#) object.

Example

This example sets the height and width of the first inline shape in the active document to 150 percent of the shape's original height and width.

```
With ActiveDocument.InlineShapes(1)  
    .ScaleHeight = 150  
    .ScaleWidth = 150  
End With
```



ScaleWidth Property

-
Scales the width of the specified inline shape relative to its original size.
Read/write **Single**.

expression.**ScaleWidth**

expression Required. An expression that returns an [InlineShape](#) object.

Example

This example sets the height and width of the first inline shape in the active document to 150 percent of the shape's original height and width.

```
With ActiveDocument.InlineShapes(1)  
    .ScaleHeight = 150  
    .ScaleWidth = 150  
End With
```



Scaling Property

Returns or sets the scaling percentage applied to the font. This property stretches or compresses text horizontally as a percentage of the current size (the scaling range is from 1 through 600). Read/write **Long**.

expression.**Scaling**

expression Required. An expression that returns a [Font](#) object.

Example

This example horizontally stretches the text in the active document to 110 percent of its original size.

```
ActiveDocument.Content.Font.Scaling = 110
```

This example compresses the text in the first paragraph in Sales.doc to 90 percent of its original size.

```
With Documents("Sales.doc").Paragraphs(1).Range.Font  
    .Scaling = 90  
    .Bold = False  
End With
```



Scope Property

Returns a [Range](#) object that represents the range of text marked by the specified comment.

expression.**Scope**

expression Required. An expression that returns a [Comment](#) object.

Example

This example displays the text associated with the first comment in the selection.

```
If Selection.Comments.Count >= 1 Then
    Set myRange = Selection.Comments(1).Scope
    MsgBox myRange.Text
End If
```

This example copies the text associated with the last comment in the active document.

```
total = ActiveDocument.Comments.Count
If total >= 1 Then ActiveDocument.Comments(total).Scope.Copy
```



↳ [Show All](#)

ScreenSize Property

Returns or sets the ideal minimum screen size (width by height, in pixels) that you should use when viewing the saved document in a Web browser. Read/write [MsoScreenSize](#).

MsoScreenSize can be one of these MsoScreenSize constants.

msoScreenSize1024x768

msoScreenSize1152x882

msoScreenSize1152x900

msoScreenSize1280x1024

msoScreenSize1600x1200

msoScreenSize1800x1440

msoScreenSize1920x1200

msoScreenSize544x376

msoScreenSize640x480

msoScreenSize720x512

msoScreenSize800x600 *default*

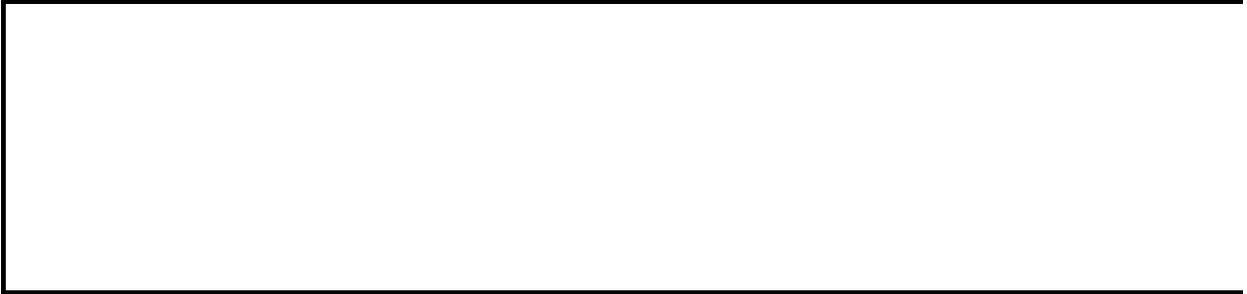
expression.**ScreenSize**

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

Example

This example sets the target screen size at 800x600 pixels.

```
Application.DefaultWebOptions.ScreenSize = _  
    msoScreenSize800x600
```



ScreenTip Property

Returns or sets the text that appears as a ScreenTip when the mouse pointer is positioned over the specified hyperlink. Read/write **String**.

expression.**ScreenTip**

expression Required. An expression that returns a [Hyperlink](#) object.

Example

This example sets the ScreenTip text for the first hyperlink in the active document.

```
ActiveDocument.Hyperlinks(1).ScreenTip = _  
    "Home"
```



ScreenUpdating Property

-
True if screen updating is turned on. Read/write **Boolean**.

expression.**ScreenUpdating**

expression Required. An expression that returns an [Application](#) object.

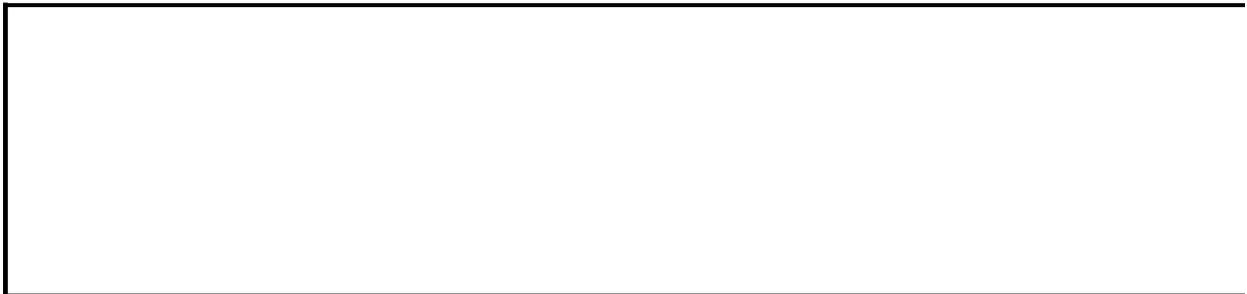
Remarks

The **ScreenUpdating** property controls most display changes on the monitor while a procedure is running. When screen updating is turned off, toolbars remain visible and Word still allows the procedure to display or retrieve information using status bar prompts, input boxes, dialog boxes, and message boxes. You can increase the speed of some procedures by keeping screen updating turned off. You must set the **ScreenUpdating** property to **True** when the procedure finishes or when it stops after an error.

Example

This example turns off screen updating and then adds a new document. Five hundred lines of text are added to the document. At every fiftieth line, the macro selects the line and refreshes the screen.

```
Application.ScreenUpdating = False
Documents.Add
For x = 1 To 500
    With ActiveDocument.Content
        .InsertAfter "This is line " & x & "."
        .InsertParagraphAfter
    End With
    If x Mod 50 = 0 Then
        ActiveDocument.Paragraphs(x).Range.Select
        Application.ScreenRefresh
    End If
Next x
Application.ScreenUpdating = True
```



Script Property

Returns a [Script](#) object, which represents a block of script or code on the specified Web page. If the page contains no script, nothing is returned.

expression.**Script**

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

Example

This example displays the type of scripting language used in the first shape in the active document.

```
Set objScr = ActiveDocument.Shapes(1).Script
If Not (objScr Is Nothing) Then
    Select Case objScr.Language
        Case msoScriptLanguageVisualBasic
            MsgBox "VBScript"
        Case msoScriptLanguageJava
            MsgBox "JavaScript"
        Case msoScriptLanguageASP
            MsgBox "Active Server Pages"
        Case Else
            MsgBox "Other scripting language"
    End Select
End If
```



Scripts Property

Returns a [Scripts](#) collection that represents the collection of HTML scripts in the specified object.

expression.**Scripts**

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

Example

This example displays the text in the first [Script](#) object of the active document.

```
Debug.Print ActiveDocument.Scripts(1).ScriptText
```

This example tests the second **Script** object in the specified range to determine its language.

```
Select Case Selection.Range.Scripts(2).Language  
    Case msoScriptLanguageASP  
        MsgBox "Active Server Pages"  
    Case msoScriptLanguageVisualBasic  
        MsgBox "VBScript"  
    Case msoScriptLanguageJava  
        MsgBox "JavaScript"  
    Case msoScriptLanguageOther  
        MsgBox "Unknown type of script"  
End Select
```



↳ [Show All](#)

SectionDirection Property

Returns or sets the reading order and alignment for the specified sections.
Read/write [WdSectionDirection](#).

WdSectionDirection can be one of these WdSectionDirection constants.

wdSectionDirectionLtr Displays the section with left alignment and left-to-right reading order.

wdSectionDirectionRtl Displays the section with right alignment and right-to-left reading order.

expression.**SectionDirection**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the direction of the first section in the active document to right-to-left.

```
ActiveDocument.Sections(1).PageSetup.SectionDirection = _  
    wdSectionDirectionRtl
```



Sections Property

-
Returns a [Sections](#) collection that represents the sections in the specified document, range, or selection. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

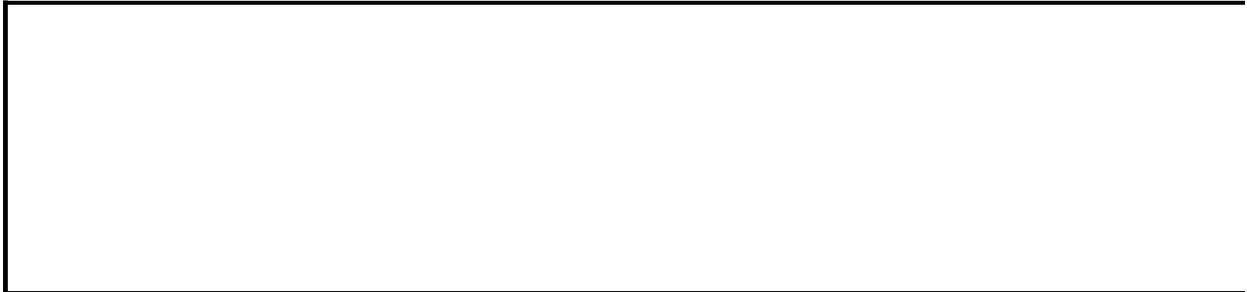
Example

This example sets the page orientation for all the sections in the active document.

```
For Each sec In ActiveDocument.Sections  
    sec.PageSetup.Orientation = wdOrientLandscape  
Next sec
```

This example creates a new document then adds some text to the document. It then creates a new section in the document and inserts text into the new section.

```
Set myDoc = Documents.Add  
Selection.InsertAfter "This is section 1."  
Set mysec = myDoc.Sections.Add  
mysec.Range.InsertAfter "This is section 2"
```



↳ [Show All](#)

SectionStart Property

Returns or sets the type of section break for the specified object. Read/write [WdSectionStart](#).

WdSectionStart can be one of these WdSectionStart constants.

wdSectionContinuous

wdSectionEvenPage

wdSectionNewColumn

wdSectionNewPage

wdSectionOddPage

expression.**SectionStart**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example changes the type of section break to continuous for all sections in the active document.

```
ActiveDocument.PageSetup.SectionStart = wdSectionContinuous
```

This example returns the type of section break used at the beginning of the second section in MyDoc.doc and applies it to all the sections in the active document.

```
mytype = Documents("MyDoc.doc").Sections(2).PageSetup.SectionStart  
ActiveDocument.PageSetup.SectionStart = mytype
```



↳ [Show All](#)

SeekView Property

Returns or sets the document element displayed in print layout view. Read/write [WdSeekView](#).

WdSeekView can be one of these WdSeekView constants.

wdSeekCurrentPageFooter
wdSeekCurrentPageHeader
wdSeekEndnotes
wdSeekEvenPagesFooter
wdSeekEvenPagesHeader
wdSeekFirstPageFooter
wdSeekFirstPageHeader
wdSeekFootnotes
wdSeekMainDocument
wdSeekPrimaryFooter
wdSeekPrimaryHeader

expression.**SeekView**

expression Required. An expression that returns a [View](#) object.

Remarks

This property generates an error if the view is not print layout view.

Example

If the active document has footnotes, this example displays footnotes in print layout view.

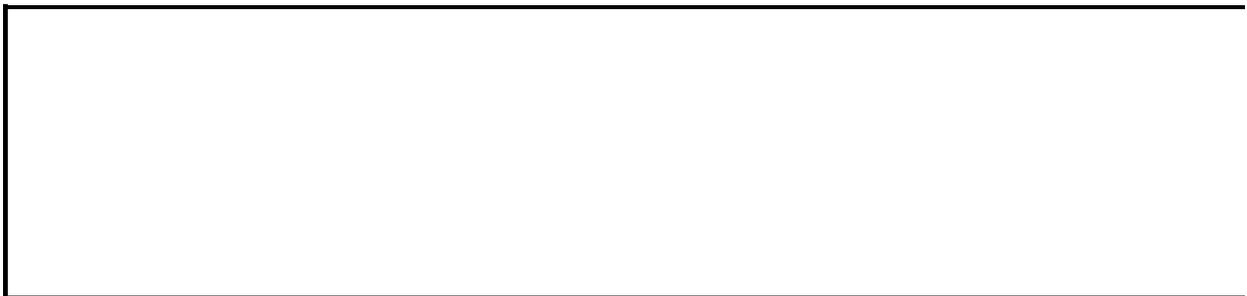
```
If ActiveDocument.Footnotes.Count >= 1 Then
    With ActiveDocument.ActiveWindow.View
        .Type = wdPrintView
        .SeekView = wdSeekFootnotes
    End With
End If
```

This example shows the first page footer for the current section.

```
ActiveDocument.PageSetup.DifferentFirstPageHeaderFooter = True
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .SeekView = wdSeekFirstPageFooter
End With
```

If the selection is in a footnote or endnote area in print layout view, this example switches to the main document.

```
Set myView = ActiveDocument.ActiveWindow.View
If myView.SeekView = wdSeekFootnotes Or _
    myView.SeekView = wdSeekEndnotes Then
    myView.SeekView = wdSeekMainDocument
End If
```



↳ [Show All](#)

SegmentType Property

Returns a value that indicates whether the segment associated with the specified node is straight or curved. Read-only [MsoSegmentType](#).

MsoSegmentType can be one of these MsoSegmentType constants.

msoSegmentCurve

msoSegmentLine

expression.**SegmentType**

expression Required. An expression that returns a [ShapeNode](#) object.

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 shape three on myDocument. Shape three must be a freeform drawing.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3).Nodes
    n = 1
    While n <= .Count
        If .Item(n).SegmentType = msoSegmentLine Then
            .SetSegmentType n, msoSegmentCurve
        End If
        n = n + 1
    Wend
End With
```



Selection Property

-

Returns the [Selection](#) object that represents a selected range or the insertion point. Read-only.

Example

This example displays the selected text.

```
If Selection.Type = wdSelectionNormal Then MsgBox Selection.Text
```

This example copies the selection from window one to the next window.

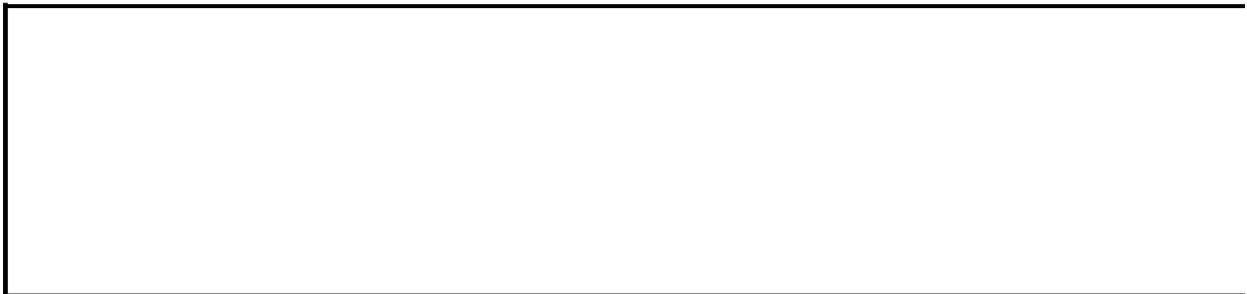
```
If Windows.Count >= 2 Then  
    Windows(1).Selection.Copy  
    Windows(1).Next.Activate  
    Selection.Paste  
End If
```

This example applies the Arial font and bold formatting to the selection.

```
With Selection.Font  
    .Bold = True  
    .Italic = False  
    .Name = "Arial"  
End With
```

If the insertion point isn't located in a table, the selection is moved to the next table.

```
If Selection.Information(wdWithInTable) = False Then  
    Selection.GoToNext What:=wdGoToTable  
End If
```



SenderCity Property

-
Returns or sets the sender's city. Not used in the U.S. English version of Microsoft Word. Read/write **String**.

expression.**SenderCity**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

SenderCode Property

Returns or sets the sender code. Not used in the U.S. English version of Microsoft Word. Read/write **String**.

expression.**SenderCode**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

SenderCompany Property

Returns or sets the company name of the person creating a letter with the Letter Wizard. Read/write **String**.

expression.**SenderCompany**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example retrieves the Letter Wizard elements from the active document. If the sender's company name isn't blank, the example displays the text in a message box.

```
If ActiveDocument.GetLetterContent.SenderCompany <> "" Then  
    MsgBox ActiveDocument.GetLetterContent.SenderCompany  
End If
```



↳ [Show All](#)

SenderGender Property

-
Returns or sets the gender used with the salutation. Not used in the U.S. English version of Microsoft Word. Read/write [WdSalutationGender](#).

WdSalutationGender can be one of these WdSalutationGender constants.

wdGenderFemale

wdGenderMale

wdGenderNeutral

wdGenderUnknown

expression.**SenderGender**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

SenderInitials Property

Returns or sets the initials of the person creating a letter with the Letter Wizard.
Read/write **String**.

expression.**SenderInitials**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object with the sender name and initials from the **User Information** tab in the **Options** dialog box (**Tools** menu). The example creates a new document and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
With myContent
    .SenderName = Application.UserName
    .SenderInitials = Application.UserInitials
End With
Documents.Add.RunLetterWizard _
    LetterContent:=myContent, WizardMode:=True
```



SenderJobTitle Property

Returns or sets the job title of the person creating a letter with the Letter Wizard.
Read/write **String**.

expression.**SenderJobTitle**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example retrieves the Letter Wizard elements from the active document and displays the sender's job title.

```
Set myLetterContent = ActiveDocument.GetLetterContent  
MsgBox myLetterContent.SenderJobTitle
```



SenderName Property

Returns or sets the name of the person creating a letter with the Letter Wizard.
Read/write **String**.

expression.**SenderName**

expression Required. An expression that returns a [LetterContent](#) object.

Example

This example creates a new [LetterContent](#) object, with the sender name and initials from the **User Information** tab in the **Options** dialog box (**Tools** menu). The example creates a new document and then runs the Letter Wizard by using the [RunLetterWizard](#) method.

```
Set myContent = New LetterContent
With myContent
    .SenderName = Application.UserName
    .SenderInitials = Application.UserInitials
End With
Documents.Add.RunLetterWizard _
    LetterContent:=myContent, WizardMode:=True
```



SenderNamefromLeft Property

Returns or sets a **Single** that represents the position, measured in points, of the sender's name from the left edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**SenderNamefromLeft**

expression Required. An expression that returns an [Envelope](#) object.

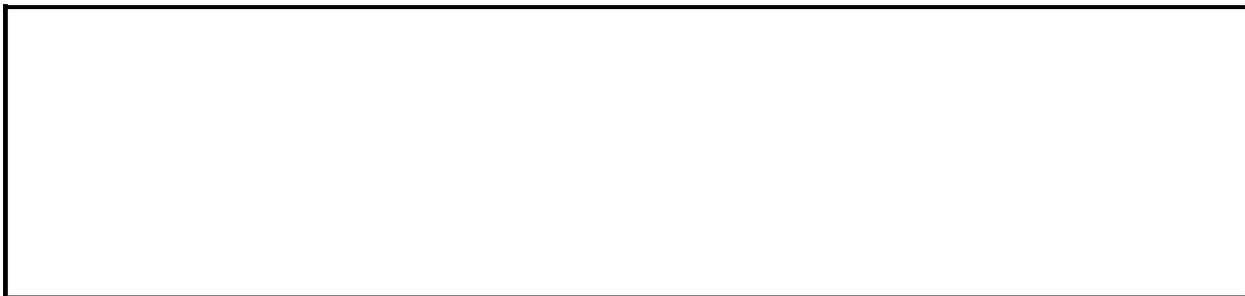
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



SenderNamefromTop Property

Returns or sets a **Single** that represents the position, measured in points, of the sender's name from the top edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**SenderNamefromTop**

expression Required. An expression that returns an [Envelope](#) object.

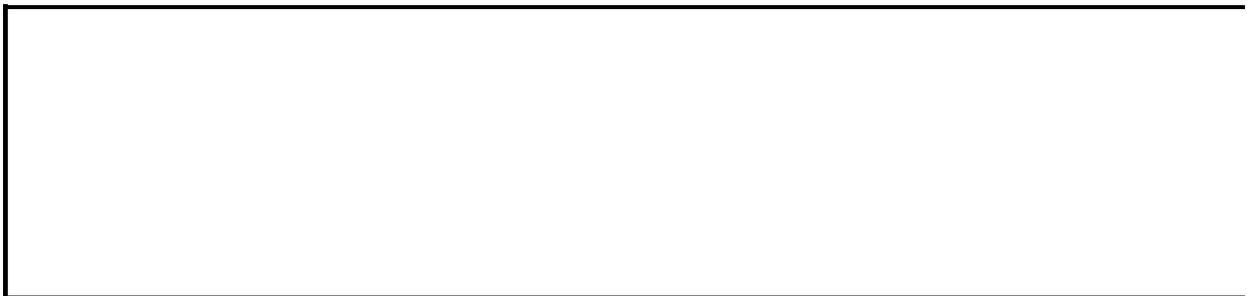
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



SenderPostalfromLeft Property

Returns or sets a **Single** that represents the position, measured in points, of the sender's postal code from the left edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**SenderPostalfromLeft**

expression Required. An expression that returns an [Envelope](#) object.

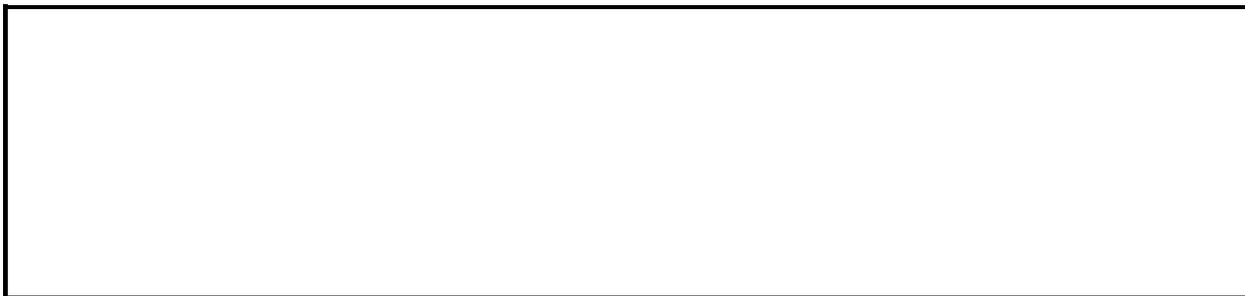
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



SenderPostalfromTop Property

Returns or sets a **Single** that represents the position, measured in points, of the sender's postal code from the top edge of the envelope. Used for Asian language envelopes. Read/write.

expression.**SenderPostalfromTop**

expression Required. An expression that returns an [Envelope](#) object.

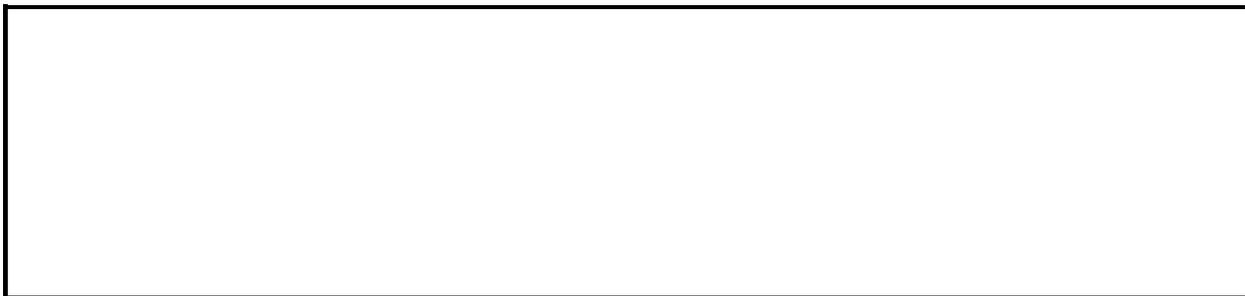
Remarks

For more information on using Microsoft Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks that the active document is a mail merge envelope and that it is formatted for vertical type. If so, it positions the recipient and sender address information.

```
Sub NewEnvelopeMerge()  
  With ActiveDocument  
    If .MailMerge.MainDocumentType = wdEnvelopes Then  
      With ActiveDocument.Envelope  
        If .Vertical = True Then  
          .RecipientNamefromLeft = InchesToPoints(2.5)  
          .RecipientNamefromTop = InchesToPoints(2)  
          .RecipientPostalfromLeft = InchesToPoints(1.5)  
          .RecipientPostalfromTop = InchesToPoints(0.5)  
          .SenderNamefromLeft = InchesToPoints(0.5)  
          .SenderNamefromTop = InchesToPoints(2)  
          .SenderPostalfromLeft = InchesToPoints(0.5)  
          .SenderPostalfromTop = InchesToPoints(3)  
        End If  
      End With  
    End If  
  End With  
End Sub
```



SenderReference Property

-
Not used in the U.S. English version of Microsoft Word. Read/write **String**.

expression.**SenderReference**

expression Required. An expression that returns a [LetterContent](#) object.

Remarks

This property may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

--

SendMailAttach Property

-

True if the **Send To** command on the **File** menu inserts the active document as an attachment to a mail message. **False** if the **Send To** command inserts the contents of the active document as text in a mail message. Read/write **Boolean**.

expression.**SendMailAttach**

expression Required. An expression that returns an [Options](#) object.

Example

This example opens a new mail message that has the active document attached to it.

```
Options.SendMailAttach = True  
ActiveDocument.SendMail
```

This example returns the state of the **Mail as attachment** option on the **General** tab of the **Options** dialog box.

```
Msgbox Options.SendMailAttach
```



Sentences Property

-
Returns a [Sentences](#) collection that represents all the sentences in the range, selection, or document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example copies the first sentences in the active document.

```
ActiveDocument.Sentences(1).Copy
```

This example deletes the last sentence in the active document.

```
ActiveDocument.Sentences.Last.Delete
```

This example displays the number of sentences in the first paragraph in the active document.

```
MsgBox ActiveDocument.Paragraphs(1).Range _  
    .Sentences.Count & " sentences"
```



↳ [Show All](#)

Separator Property

▶ [Separator property as it applies to the **CaptionLabel** object.](#)

Returns or sets the character between the chapter number and the sequence number. Read/write [WdSeparatorType](#).

WdSeparatorType can be one of these WdSeparatorType constants.

wdSeparatorColon

wdSeparatorEnDash

wdSeparatorPeriod

wdSeparatorEmDash

wdSeparatorHyphen

expression.**Separator**

expression Required. An expression that returns a [CaptionLabel](#) object.

▶ [Separator property as it applies to the **Endnotes** and **Footnotes** objects.](#)

Returns a [Range](#) object that represents the endnote or footnote separator.

expression.**Separator**

expression Required. An expression that returns one of the above objects.

▶ [Separator property as it applies to the **TableOfAuthorities** object.](#)

Returns or sets the characters (up to five) between the sequence number and the page number. A hyphen (-) is the default character. This property corresponds to the \d switch for a Table of Authorities (TOA) field. Read/write **String**.

expression.**Separator**

expression Required. An expression that returns a [TableOfAuthorities](#) object.

Example

▶ [As applies to the **CaptionLabel** object.](#)

This example inserts a Figure caption that has a colon (:) between the chapter number and the sequence number.

```
With CaptionLabels("Figure")
    .Separator = wdSeparatorColon
    .IncludeChapterNumber = True
End With
Selection.InsertCaption "Figure"
```

▶ [As applies to the **Footnotes** object.](#)

This example changes the footnote separator to a single border indented 3 inches from the right margin.

```
With ActiveDocument.Footnotes.Separator
    .Delete
    .Borders(wdBorderTop).LineStyle = wdLineStyleSingle
    .ParagraphFormat.RightIndent = InchesToPoints(3)
End With
```

▶ [As applies to the **TableOfAuthorities** object.](#)

This example inserts a table of authorities at the beginning of the active document, and then it formats the table to include a sequence number and a page number, separated by a hyphen (-).

```
Set myRange = ActiveDocument.Range(0, 0)
With ActiveDocument.TablesOfAuthorities.Add(Range:=myRange)
    .IncludeSequenceName = "Chapter"
    .Separator = "-"
End With
```



SequenceCheck Property

True to check the sequence of independent characters for South Asian text.
Read/write **Boolean**.

expression.**SequenceCheck**

expression Required. An expression that returns an [Options](#) object.

Example

This example enables sequence checking, allowing the user to type a valid sequence of independent characters to form valid character cells in South Asian text.

```
Sub CheckSequence()  
    Options.SequenceCheck = True  
End Sub
```



Shaded Property

True if shading is applied to form fields. Read/write **Boolean**.

expression.**Shaded**

expression Required. An expression that returns a [FormFields](#) collection object.

Remarks

Shading makes form fields easier to locate in a document and doesn't affect the printed output.

Example

This example removes shading from form fields in Employment Form.doc.

```
Documents("Employment Form.doc").FormFields.Shaded = False
```

This example adds shading to the form fields in the active document and protects the document for forms.

```
With ActiveDocument  
    .FormFields.Shaded = True  
    .Protect Type:=wdAllowOnlyFormFields, NoReset:=True  
End With
```



Shading Property

Returns a [Shading](#) object that refers to the shading formatting for the specified object.

expression.**Shading**

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

Example

This example applies yellow shading to the first paragraph in the selection.

```
With Selection.Paragraphs(1).Shading  
    .Texture = wdTexture12Pt5Percent  
    .BackgroundPatternColorIndex = wdYellow  
    .ForegroundPatternColorIndex = wdBlack  
End With
```

This example applies horizontal line texture to the first row in table one.

```
If ActiveDocument.Tables.Count >= 1 Then  
    With ActiveDocument.Tables(1).Rows(1).Shading  
        .Texture = wdTextureHorizontal  
    End With  
End If
```

This example applies 10 percent shading to the first word in the active document.

```
ActiveDocument.Words(1).Shading.Texture = wdTexture10Percent
```



↳ [Show All](#)

Shadow Property

▶ [Shadow property as it applies to the **Borders** object.](#)

True if the specified border is formatted as shadowed. Read/write **Boolean**.

expression.**Shadow**

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

▶ [Shadow property as it applies to the **Font** object.](#)

True if the specified font is formatted as shadowed. Can be **True**, **False**, or **wdUndefined**. Read/write **Long**.

expression.**Shadow**

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

▶ [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 **Borders** object.](#)

This example demonstrates two different border styles in a new document.

```
Set myRange = Documents.Add.Content
With myRange
    .InsertAfter "Demonstration of border with shadow."
    .InsertParagraphAfter
    .InsertParagraphAfter
    .InsertAfter "Demonstration of border without shadow."
End With
With ActiveDocument
    .Paragraphs(1).Borders.Shadow = True
    .Paragraphs(3).Borders.Enable = True
End With
```

▶ [As it applies to the **Font** object.](#)

This example applies shadow and bold formatting to the selection.

```
If Selection.Type = wdSelectionNormal Then
    With Selection.Font
        .Shadow = True
        .Bold = True
    End With
Else
    MsgBox "You need to select some text."
End If
```

▶ [As it applies to the **Shape** and **ShapeRange** objects.](#)

This example adds an arrow with shadow formatting to the active document.

```
Set myShape = ActiveDocument.Shapes _
    .AddShape(Type:=msoShapeRightArrow, _
    Left:=90, Top:=79, Width:=64, Height:=43)
myShape.Shadow.Type = msoShadow5
```



Shape Property

Returns a [Shape](#) object for the specified hyperlink or diagram node.

expression.**Shape**

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

Remarks

If a hyperlink isn't represented by a shape, an error occurs.

Example

This example changes the fill color for the shape that represents the first hyperlink in the active document. For this example to work, the hyperlink must be represented by a shape.

```
ActiveDocument.Hyperlinks(1).Shape.Fill.ForeColor.RGB = _  
    RGB(255, 255, 0)
```



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. Read-only.

Example

The following example sets the fill foreground color to purple for all the shapes in the active document.

```
ActiveDocument.Content.ShapeRange.Fill.ForeColor.RGB = _  
    RGB(255, 0, 255)
```

The following example applies shadow formatting to all the shapes in the selection.

```
Selection.ShapeRange.Shadow.Type = msoShadow6
```



Shapes Property

Returns a [Shapes](#) collection that represents all the **Shape** objects in the specified document, header, or footer. This collection can contain drawings, shapes, pictures, OLE objects, ActiveX controls, text objects, and callouts. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Remarks

The **Shapes** property, when applied to a document, returns all the **Shape** objects in the main story of the document, excluding the headers and footers. When applied to a **HeaderFooter** object, the **Shapes** property returns all the **Shape** objects found in all the headers and footers in the document.

Example

This example creates a new document, adds a rectangle to it that's 100 points wide and 50 points high, and sets the upper-left corner of the rectangle to be 5 points from the left edge and 25 points from the upper-left corner of the page.

```
Set myDoc = Documents.Add  
myDoc.Shapes.AddShape msoShapeRectangle, 5, 25, 100, 50
```

This example sets the fill texture for all the shapes in the active document.

```
For each s in ActiveDocument.Shapes  
    s.Fill.PresetTextured msoTextureOak  
Next s
```

This example adds a shadow to the first shape in the active document.

```
Set myShape = ActiveDocument.Shapes(1)  
myShape.Shadow.Type = msoShadow6
```

This example displays a count of all the shapes in the primary header and footer of the first section of the active document.

```
MsgBox ActiveDocument.Sections(1). _  
    Headers(wdHeaderFooterPrimary).Shapes.Count
```



ShowAll Property

-
True if all nonprinting characters (such as hidden text, tab marks, space marks, and paragraph marks) are displayed. Read/write **Boolean**.

expression.**ShowAll**

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

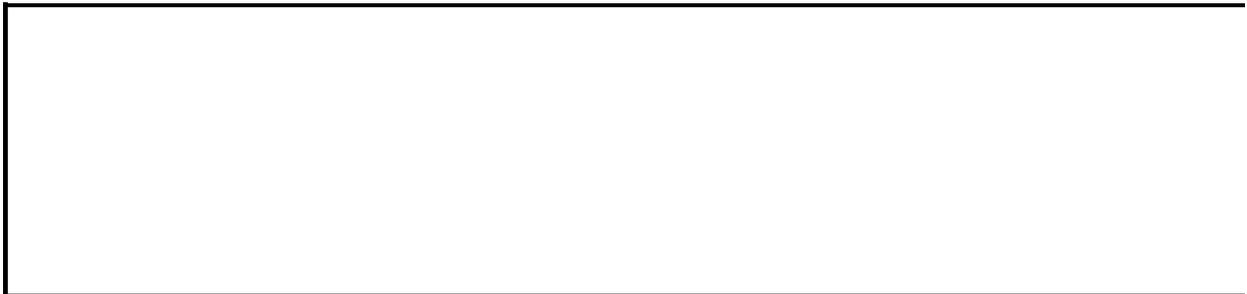
Example

This example displays all nonprinting characters in the active window.

```
ActiveDocument.ActiveWindow.View.ShowAll = True
```

This example toggles the display of nonprinting characters in the first window.

```
Windows(1).View.ShowAll = Not Windows(1).View.ShowAll
```



ShowAnimation Property

True if text animation is displayed. Read/write **Boolean**.

expression.**ShowAnimation**

expression Required. An expression that returns a [View](#) object.

Example

This example turns on text animation in the active window and then applies sparkle-text animation to the selection.

```
ActiveDocument.ActiveWindow.View.ShowAnimation = True  
Selection.Font.Animation = wdAnimationSparkleText
```

This example turns off font animation in all open windows.

```
For Each aWindow In Windows  
    aWindow.View.ShowAnimation = False  
Next aWindow
```



ShowBookmarks Property

True if square brackets are displayed at the beginning and end of each bookmark. Read/write **Boolean**.

expression.**ShowBookmarks**

expression Required. An expression that returns a [View](#) object.

Example

This example displays square brackets around bookmarks in all windows.

```
For Each aWindow In Windows
    aWindow.View.ShowBookmarks = True
Next aWindow
```

This example marks the selection with a bookmark, displays square brackets around each bookmark in the active document, and then collapses the selection.

```
ActiveDocument.Bookmarks.Add Range:=Selection.Range, Name:="temp"
ActiveDocument.ActiveWindow.View.ShowBookmarks = True
Selection.Collapse Direction:=wdCollapseStart
```



ShowBy Property

-

Returns or sets the name of the reviewer whose comments are shown in the comments pane. You can choose to show comments either by a single reviewer or by all reviewers. Read/write **String**.

expression.**ShowBy**

expression Required. An expression that returns a [Comments](#) collection object.

Remarks

To view the comments by all reviewers, set this property to "All Reviewers."

Example

The following example displays comments made by Don Funk.

```
If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.ActiveWindow.View.SplitSpecial = wdPaneComments
    ActiveDocument.Comments.ShowBy = "Don Funk"
End If
```



ShowCodes Property

True if field codes are displayed for the specified field instead of field results.
Read/write **Boolean**.

expression.**ShowCodes**

expression Required. An expression that returns a [Field](#) object.

Example

This example selects the next field and displays the field codes.

```
With Selection
    .GoTo What:=wdGoToField
    .Expand Unit:=wdWord
    If .Fields.Count = 1 Then .Fields(1).ShowCodes = True
End With
```

This example updates and displays the result of the first field in the active document.

```
If ActiveDocument.Fields.Count >= 1 Then
    With ActiveDocument.Fields(1)
        .Update
        .ShowCodes = False
    End With
End If
```



ShowComments Property

True for Microsoft Word to display the comments in a document. Read/write **Boolean**.

expression.**ShowComments**

expression Required. An expression that returns a [View](#) object.

Remarks

If revision marks are displayed in balloons in the right or left margin, comments are displayed in balloons. If revision marks are displayed inline, the text to which comments apply is surrounded by wide-angled square brackets; when a user places the mouse pointer over the text within the brackets, the related comment is displayed in a square balloon directly above the mouse pointer.

Example

This example hides the comments in the active document. This example assumes that the document in the active window contains one or more comments.

```
Sub HideComments()  
    ActiveWindow.View.ShowComments = False  
End Sub
```



ShowControlCharacters Property

True if bidirectional control characters are visible in the current document.
Read/write **Boolean**.

expression.**ShowControlCharacters**

expression Required. An expression that returns an [Options](#) object.

Example

This example hides bidirectional control characters in the current document.

```
Options.ShowControlCharacters = False
```



ShowDiacritics Property

-
True if diacritics are visible in a right-to-left language document. Read/write **Boolean**.

expression.**ShowDiacritics**

expression Required. An expression that returns an [Options](#) object.

Example

This example hides diacritics in the current document.

```
Options.ShowDiacritics = False
```



ShowDrawings Property

True if objects created with the drawing tools are displayed in print layout view.
Read/write **Boolean**.

expression.**ShowDrawings**

expression Required. An expression that returns a [View](#) object.

Example

This example switches the active window to print layout view and displays objects created with the drawing tools.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdPrintView  
    .ShowDrawings = True  
End With
```



ShowFieldCodes Property

True if field codes are displayed. Read/write **Boolean**.

expression.**ShowFieldCodes**

expression Required. An expression that returns a [View](#) object.

Example

This example hides field codes in the window for Document1.

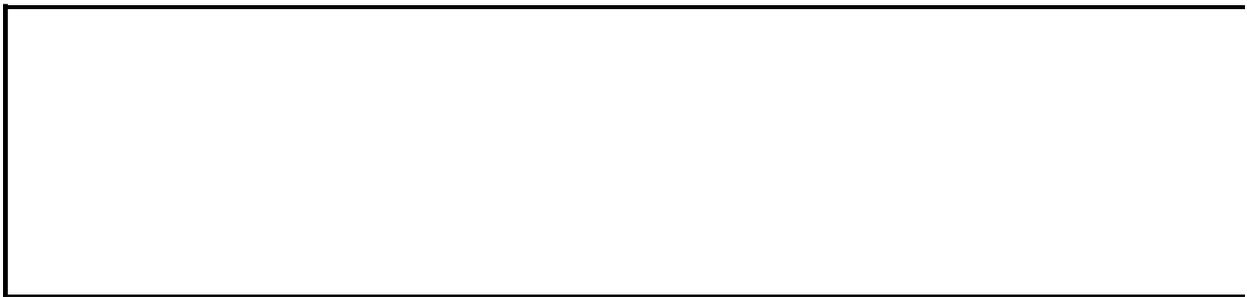
```
Windows("Document1").View.ShowFieldCodes = False
```

This example shows field codes in the first window.

```
Windows(1).View.ShowFieldCodes = True
```

This example toggles field codes in the active window.

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = _  
    Not ActiveDocument.ActiveWindow.View.ShowFieldCodes
```



ShowFirstLineOnly Property

True if only the first line of body text is shown in outline view. Read/write **Boolean**.

expression.**ShowFirstLineOnly**

expression Required. An expression that returns a [View](#) object.

Remarks

This property generates an error if the view isn't outline or master document view.

Example

This example switches the active window to outline view and hides all but the first line of body text.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdOutlineView  
    .ShowFirstLineOnly = True  
End With
```



ShowFirstPageNumber Property

True if the page number appears on the first page in the section. Read/write **Boolean**.

expression.**ShowFirstPageNumber**

expression Required. An expression that returns a [PageNumbers](#) collection object.

Remarks

Setting this property to **True** automatically adds page numbers to a section.

Example

This example checks to see whether the page number appears on the first page in the active document.

```
Set myDoc = ActiveDocument
first = myDoc.Sections(1).Headers(wdHeaderFooterPrimary). _
    PageNumbers.ShowFirstPageNumber
Msgbox "This document shows numbers on the first page - " & first
```

This example adds page numbers to the active document.

```
ActiveDocument.Sections(1) _
    .Headers(wdHeaderFooterPrimary).PageNumbers _
    .ShowFirstPageNumber = True
```



ShowFormat Property

True if character formatting is visible in outline view. Read/write **Boolean**.

expression.**ShowFormat**

expression Required. An expression that returns a [View](#) object.

Remarks

This property generates an error if the view isn't outline or master document view.

Example

This example switches the active window to outline view and shows character formatting.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdOutlineView  
    .ShowFormat = True  
End With
```



ShowFormatChanges Property

True for Microsoft Word to display formatting changes made to a document with Track Changes enabled. Read/write **Boolean**.

expression.**ShowFormatChanges**

expression Required. An expression that returns a [View](#) object.

Example

This example hides the formatting changes made to the active document. This example assumes that formatting changes have been made to a document in which Track Changes is enabled.

```
Sub HideFormattingChanges()  
    ActiveWindow.View.ShowFormatChanges = False  
End Sub
```



ShowFormatError Property

-

True for Microsoft Word to mark inconsistencies in formatting by placing a squiggly underline beneath text formatted similarly to other formatting that is used more frequently in a document. Read/write **Boolean**.

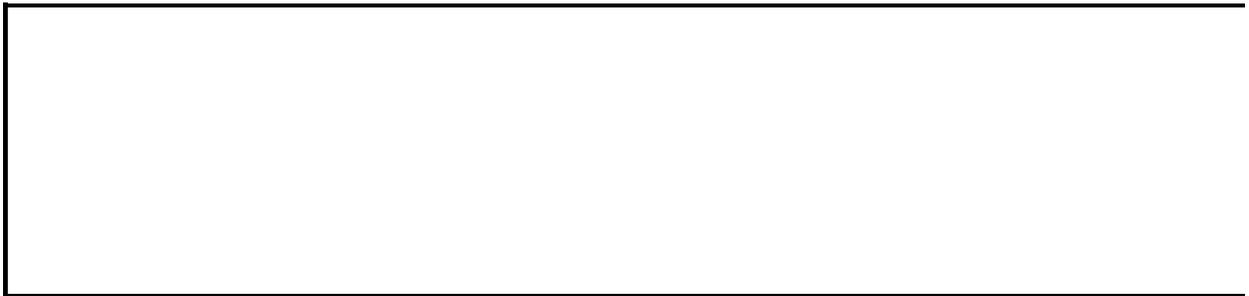
expression.**ShowFormatError**

expression Required. An expression that returns an [Options](#) object.

Example

This example enables Word to keep track of formatting in documents but does not display a squiggly underline beneath text.

```
Sub ShowFormatErrors()  
  
    With Options  
        .FormatScanning = True 'Enables keeping track of formatting  
        .ShowFormatError = False  
    End With  
  
End Sub
```



ShowGrammaticalErrors Property

-
True if grammatical errors are marked by a wavy green line in the specified document. Read/write **Boolean**.

Note To view grammatical errors in your document, you must set the **CheckGrammarAsYouType** property to **True**.

Example

This example sets Word to check for grammatical errors as you type and to display any errors found in the active document.

```
Options.CheckGrammarAsYouType = True  
ActiveDocument.ShowGrammaticalErrors = True
```



ShowHidden Property

-
True if hidden bookmarks are included in the **Bookmarks** collection. This property also controls whether hidden bookmarks are listed in the **Bookmark** dialog box (**Insert** menu). Read/write **Boolean**.

expression.**ShowHidden**

expression Required. An expression that returns a [Bookmarks](#) collection object.

Remarks

Hidden bookmarks are automatically inserted when cross-references are inserted into the document.

Example

This example displays the **Bookmark** dialog box with both visible and hidden bookmarks listed.

```
ActiveDocument.Bookmarks.ShowHidden = True  
Dialogs(wdDialogInsertBookmark).Show
```

This example displays the name of each hidden bookmark in the document. Hidden bookmarks in a Word document begin with an underscore (_).

```
ActiveDocument.Bookmarks.ShowHidden = True  
For Each aBookmark In ActiveDocument.Bookmarks  
    If Left(aBookmark.Name, 1) = "_" Then MsgBox aBookmark.Name  
Next aBookmark
```



ShowHiddenText Property

True if text formatted as hidden text is displayed. Read/write **Boolean**.

expression.**ShowHiddenText**

expression Required. An expression that returns a [View](#) object.

Example

This example hides text formatted as hidden text in each window.

```
For Each myWindow In Windows
    myWindow.View.ShowHiddenText = False
Next myWindow
```

This example toggles the display of hidden text.

```
ActiveDocument.ActiveWindow.View.ShowHiddenText = _
    Not ActiveDocument.ActiveWindow.View.ShowHiddenText
```



ShowHighlight Property

True if highlight formatting is displayed and printed with a document.
Read/write **Boolean**.

expression.**ShowHighlight**

expression Required. An expression that returns a [View](#) object.

Example

This example toggles the display of highlighting in the active document.

```
ActiveDocument.ActiveWindow.View.ShowHighlight = _  
    Not ActiveDocument.ActiveWindow.View.ShowHighlight
```

This example prints the active document without highlight formatting.

```
With ActiveDocument  
    .ActiveWindow.View.ShowHighlight = False  
    .PrintOut  
End With
```



ShowHyphens Property

True if optional hyphens are displayed. An optional hyphen indicates where to break a word when it falls at the end of a line. Read/write **Boolean**.

expression.**ShowHyphens**

expression Required. An expression that returns a [View](#) object.

Example

This example inserts an optional hyphen before the selection and then displays optional hyphens in the active window.

```
Selection.InsertBefore Chr(31)  
ActiveDocument.ActiveWindow.View.ShowHyphens = True
```



ShowInsertionsAndDeletions Property

True for Microsoft Word to display insertions and deletions that were made to a document with Track Changes enabled. Read/write **Boolean**.

expression.**ShowInsertionsAndDeletions**

expression Required. An expression that returns a [View](#) object.

Example

This example hides the insertions and deletions made in a document. This example assumes that the document in the active window contains revisions made by one or more reviewers.

```
Sub HideInsertDelete()  
    ActiveWindow.View.ShowInsertionsAndDeletions = False  
End Sub
```



ShowMainTextLayer Property

True if the text in the specified document is visible when the header and footer areas are displayed. This property is equivalent to the **Show/Hide Document Text** button on the **Header and Footer** toolbar. Read/write **Boolean**.

expression.**ShowMainTextLayer**

expression Required. An expression that returns a [View](#) object.

Example

This example displays the document header in the active window and hides the document text.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdPrintView  
    .SeekView = wdSeekCurrentPageHeader  
    .ShowMainTextLayer = False  
End With
```



ShowObjectAnchors Property

True if object anchors are displayed next to items that can be positioned in print layout view. Read/write **Boolean**.

expression.**ShowObjectAnchors**

expression Required. An expression that returns a [View](#) object.

Example

This example adds a frame around the selection, switches the active window to print layout view, and shows object anchors for framed objects.

```
Selection.Frames.Add(Range:=Selection.Range).LockAnchor = True
With ActiveDocument.ActiveWindow.View
    .Type = wdPrintView
    .ShowObjectAnchors = True
End With
```



ShowOptionalBreaks Property

True if Microsoft Word displays optional line breaks. Read/write **Boolean**.

expression.**ShowOptionalBreaks**

expression Required. An expression that returns a [View](#) object.

Example

This example displays the optional line breaks in the active window.

```
ActiveDocument.ActiveWindow.View.ShowOptionalBreaks = True
```



ShowParagraphs Property

True if paragraph marks are displayed. Read/write **Boolean**.

expression.**ShowParagraphs**

expression Required. An expression that returns a [View](#) object.

Example

This example hides paragraph marks in the active window.

```
ActiveDocument.ActiveWindow.View.ShowParagraphs = False
```



ShowPicturePlaceholders Property

True if blank boxes are displayed as placeholders for pictures. Read/write **Boolean**.

expression.**ShowPicturePlaceholders**

expression Required. An expression that returns a [View](#) object.

Example

This example inserts a picture in the active document and displays picture placeholders in the active window.

```
Selection.Collapse Direction:=wdCollapseStart
ActiveDocument.InlineShapes.AddPicture Range:=Selection.Range, _
    FileName:="C:\Windows\Bubbles.bmp"
ActiveDocument.ActiveWindow.View.ShowPicturePlaceholders = True
```



ShowReadabilityStatistics Property

True if Microsoft Word displays a list of summary statistics, including measures of readability, when it has finished checking grammar. Read/write **Boolean**.

expression.**ShowReadabilityStatistics**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to show readability statistics, and then it checks the spelling and grammar in the active document.

```
Options.ShowReadabilityStatistics = True  
ActiveDocument.CheckGrammar
```

This example returns the current status of the **Show readability statistics** option on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.ShowReadabilityStatistics
```



ShowRevisions Property

-

True if tracked changes in the specified document are shown on the screen.
Read/write **Boolean**.

Example

This example sets the active document so that it tracks changes and makes them visible on the screen.

```
With ActiveDocument
    .TrackRevisions = True
    .ShowRevisions = True
End With
```



ShowRevisionsAndComments Property

True for Microsoft Word to display revisions and comments that were made to a document with Track Changes enabled. Read/write **Boolean**.

expression.**ShowRevisionsAndComments**

expression Required. An expression that returns a [View](#) object.

Example

This example hides the revisions and comments in a document. This example assumes that the document in the active window contains revisions made by one or more reviewers.

```
Sub ShowRevsComments()  
    ActiveWindow.View.ShowRevisionsAndComments = False  
End Sub
```



ShowSendToCustom Property

Returns or sets a **String** corresponding to the caption on a custom button on the **Complete the merge** step (step six) of the Mail Merge Wizard. Read/write.

expression.**ShowSendToCustom**

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

Remarks

When a user clicks the custom button, the [MailMergeWizardSendToCustom](#) event executes.

Example

This example displays a custom button on the sixth step of the Mail Merge Wizard only for mailing labels.

```
Sub ShowCustomButton()  
    With ActiveDocument.MailMerge  
        If .MainDocumentType = wdMailingLabels Then  
            .ShowSendToCustom = "Custom Label Processing"  
        End If  
    End With  
End Sub
```



ShowSpaces Property

True if space characters are displayed. Read/write **Boolean**.

expression.**ShowSpaces**

expression Required. An expression that returns a [View](#) object.

Example

This example inserts spaces before the selection and displays space characters in the active window.

```
Selection.InsertBefore "    "  
ActiveDocument.ActiveWindow.View.ShowSpaces = True
```



ShowSpellingErrors Property

-
True if Microsoft Word underlines spelling errors in the document. Read/write **Boolean**.

Remarks

To view spelling errors in a document, you must set the **CheckSpellingAsYouType** property to **True**.

Example

This example sets Word to hide the wavy red line that denotes possible spelling errors in the active document.

```
ActiveDocument.ShowSpellingErrors = False
```

This example sets Word to show spelling errors in the active document.

```
Options.CheckSpellingAsYouType = True  
ActiveDocument.ShowSpellingErrors = True
```

This example returns the current status of the **Hide spelling errors in this document** checkbox in the **Spelling** area on the **Spelling & Grammar** tab in the **Options** dialog box.

```
temp = ActiveDocument.ShowSpellingErrors
```



ShowStartupDialog Property

True to display the Task Pane when starting Microsoft Word. Read/write **Boolean**.

expression.**ShowStartupDialog**

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

Remarks

The **ShowStartupDialog** is a global option, and the new setting will take effect only after you restart Word. Use the [Visible](#) property of the **CommandBars** collection show or hide the Task Pane without restarting Word.

Example

This example turns off the Task Pane, so it won't display upon starting Word. This will not take effect until the next time the user starts Word.

```
Sub HideStartUpDlg()  
    Application.ShowStartupDialog = False  
End Sub
```



ShowSummary Property

-

True if an automatic summary is displayed for the specified document.
Read/write **Boolean**.

Example

This example hides everything in the active document except the summary text.

```
With ActiveDocument
    .SummaryViewMode = wdSummaryModeHideAllButSummary
    .SummaryLength = 30
    .ShowSummary = True
End With
```



ShowTabs Property

True if tab characters are displayed. Read/write **Boolean**.

expression.**ShowTabs**

expression Required. An expression that returns a [View](#) object.

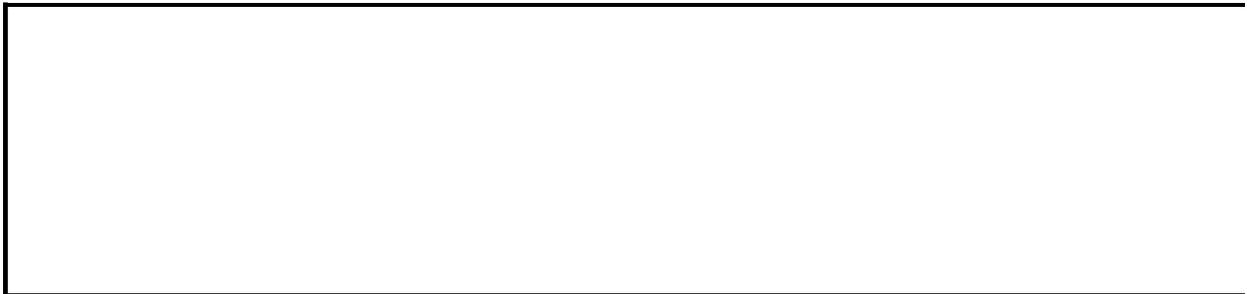
Example

This example inserts a tab before the selection and displays tab characters in the window for Document2.

```
With Windows("Document2")
    .Activate
    .View.ShowTabs = True
End With
Selection.InsertBefore vbTab
Selection.Collapse Direction:=wdCollapseEnd
```

This example splits the active window, shows tab characters in the first pane, and hides tab characters in the second pane.

```
With ActiveDocument.ActiveWindow
    .Split = True
    .Panes(1).View.ShowTabs = True
    .Panes(2).View.ShowTabs = False
End With
```



ShowTextBoundaries Property

True if dotted lines are displayed around page margins, text columns, objects, and frames in print layout view. Read/write **Boolean**.

expression.**ShowTextBoundaries**

expression Required. An expression that returns a [View](#) object.

Example

This example switches the active window to page view and displays text boundary lines.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdPrintView  
    .ShowTextBoundaries = True  
End With
```



ShowTip Property

-
True if text associated with a comment is displayed in a ScreenTip. The ScreenTip remains displayed until you click the mouse or press a key.
Read/write **Boolean**.

expression.**ShowTip**

expression Required. An expression that returns a [Comment](#) object.

Example

This example shows the ScreenTip for the first comment in the active document.

```
If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.Comments(1).ShowTip = True
End If
```

This example shows the ScreenTip for the next comment in the active document.

```
If ActiveDocument.Comments.Count >= 1 Then
    With Selection
        .GoTo What:=wdGotoComment, Which:=wdGotoNext
        .MoveEnd Unit:=wdWord, Count:=1
        .Comments(1).ShowTip = True
    End With
End If
```



ShowVisualBasicEditor Property

True if the Visual Basic Editor window is visible. Read/write **Boolean**.

expression.**ShowVisualBasicEditor**

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

Example

This example makes the Visual Basic Editor window visible.

```
Application.ShowVisualBasicEditor = True
```



ShowWindowsInTaskbar Property

-

True displays opened documents in the task bar, the default Single Document Interface (SDI). **False** lists opened documents only in the Window menu, providing the appearance of a Multiple Document Interface (MDI). Read/write **Boolean**.

expression.**ShowWindowsInTaskbar**

expression Required. An expression that returns an [Application](#) object.

Example

This example switches the interface to list open documents only on the Window menu.

```
Sub SDIToMDI()  
    Application.ShowWindowsInTaskbar = False  
End Sub
```



↳ [Show All](#)

Side Property

Returns or sets a value that indicates whether the document text should wrap on both sides of the specified shape, on either the left or right side only, or on the side of the shape that's farthest from the page margin. If the text wraps on only one side of the shape, there's a text-free area between the other side of the shape and the page margin. Read/write [WdWrapSideType](#).

WdWrapSideType can be one of these WdWrapSideType constants.

wdWrapBoth

wdWrapLargest

wdWrapLeft

wdWrapRight

expression.**Side**

expression Required. An expression that returns a [WrapFormat](#) object.

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.

```
Set myOval = ActiveDocument.Shapes.AddShape(msoShapeOval, _  
    0, 0, 200, 50)  
With myEll.WrapFormat  
    .Type = wdWrapSquare  
    .Side = wdWrapBoth  
    .DistanceTop = InchesToPoints(0.1)  
    .DistanceBottom = InchesToPoints(0.1)  
    .DistanceLeft = InchesToPoints(0.1)  
    .DistanceRight = InchesToPoints(0.1)  
End With
```



SideMargin Property

Returns or sets the side margin widths (in points) for the specified custom mailing label. Read/write **Single**.

expression.**SideMargin**

expression Required. An expression that returns a [CustomLabel](#) object.

Remarks

If this property is changed to a value that isn't valid for the specified mailing label layout, an error occurs.

Example

This example creates a custom label named "VisitorPass" and defines its layout. The left and right margins for each label are 0.75 inch.

```
Set myLabel = Application.MailingLabel.CustomLabels _  
    .Add(Name:="VisitorPass", DotMatrix:=False)  
With myLabel  
    .Height = InchesToPoints(2.17)  
    .HorizontalPitch = InchesToPoints(3.5)  
    .NumberAcross = 2  
    .NumberDown = 4  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0.75)  
    .TopMargin = InchesToPoints(0.17)  
    .VerticalPitch = InchesToPoints(2.17)  
    .Width = InchesToPoints(3.5)  
End With
```



Signatures Property

Returns a [SignatureSet](#) object that represents the digital signatures for a document.

expression.**Signatures**

expression Required. An expression that returns a [Document](#) object.

Remarks

To digitally sign Microsoft Word documents and verify other signatures in them, you will need the Microsoft CryptoAPI and a unique digital signature certificate. The CryptoAPI is installed with Microsoft Internet Explorer 4.01 and higher. You can obtain a digital signature certificate from a certification authority.

Example

This example displays the **Signatures** dialog box with which you can add a digital signature to a document.

```
Sub AddSignature  
    ActiveDocument.Signatures.Add  
End Sub
```



SingleList Property

True if the specified [ListFormat](#) object contains only one list. Read-only **Boolean**.

expression.**SingleList**

expression Required. An expression that returns a [ListFormat](#) object.

Example

This example checks the selection to see whether it only contains one list. If it does, the example applies the default numbered list template to the selection.

```
temp = Selection.Range.ListFormat.SingleList  
If temp = True Then  
    Selection.Range.ListFormat.ApplyNumberDefault  
End If
```



SingleListTemplate Property

True if the entire [List](#) or [ListFormat](#) object uses the same list template. Read-only **Boolean**.

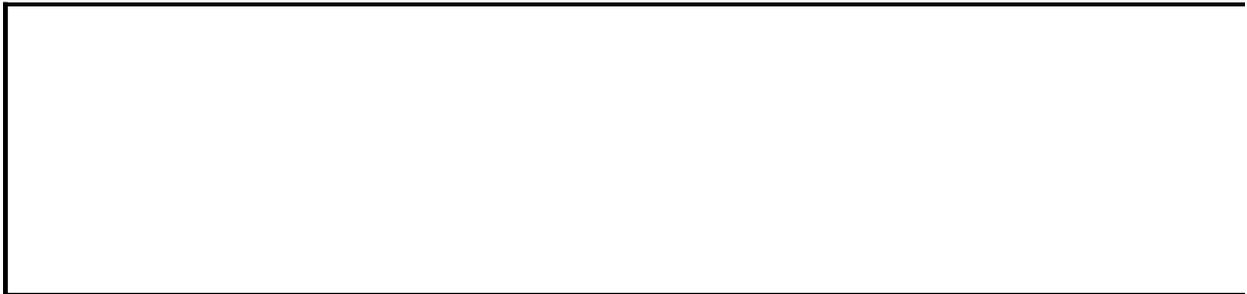
expression.**SingleListTemplate**

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

Example

This example checks to see whether the selection is formatted with a single list template. If so, the example applies the second numbered list template to the selection.

```
Set myList = Selection.Range.ListFormat
temp = myList.SingleListTemplate
If temp = True Then
    myList.ApplyListTemplate _
        ListTemplate:=ListGalleries(wdNumberGallery) _
        .ListTemplates(2)
End If
```



Size Property

Returns or sets the font size (for the [Font](#) object) or the size of the specified check box (for the [CheckBox](#) object), in points. Read/write **Single**.

expression.**Size**

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

Example

This example inserts text and then sets the font size of the seventh word of the inserted text to 20 points.

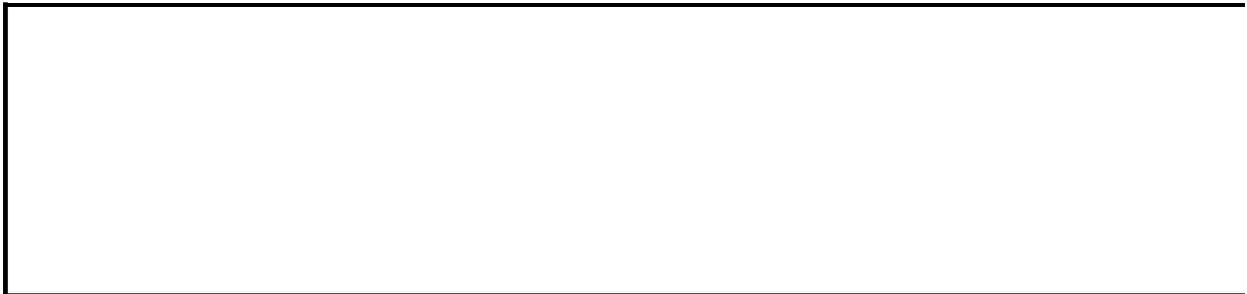
```
Selection.Collapse Direction:=wdCollapseEnd
With Selection.Range
    .Font.Reset
    .InsertBefore "This is a demonstration of font size."
    .Words(7).Font.Size = 20
End With
```

This example determines the font size of the selected text.

```
mySel = Selection.Font.Size
If mySel = wdUndefined Then
    MsgBox "There's a mix of font sizes in the selection."
Else
    MsgBox mySel & " points"
End If
```

This example sets the size of the check box named "Check1" in the active document to 14 points and then sets the check box as selected.

```
With ActiveDocument.FormFields("Check1").CheckBox
    .AutoSize = False
    .Size = 14
    .Value = True
End With
```



SizeBi Property

Returns or sets the font size in points. Read/write **Single**.

expression.**SizeBi**

expression Required. An expression that returns a [Font](#) object.

Remarks

The **SizeBi** property applies to text in a right-to-left language.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the font size of the first word to 20 points.

```
With ActiveDocument.Paragraphs(1).Range  
    .Words(1).Font.SizeBi = 20  
End With
```



SmallCaps Property

-
True if the font is formatted as small capital letters. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**SmallCaps**

expression Required. An expression that returns a [Font](#) object.

Remarks

Setting the **SmallCaps** property to **True** sets the [AllCaps](#) property to **False**, and vice versa.

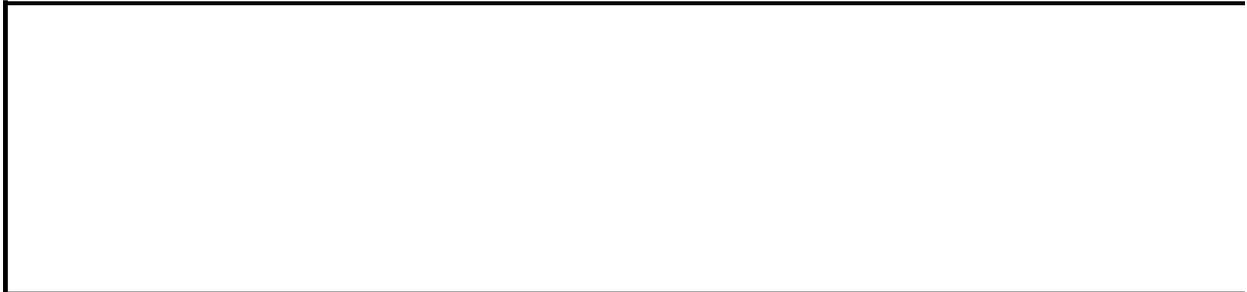
Example

This example demonstrates the difference between small capital letters and all capital letters in a new document.

```
Set myRange = Documents.Add.Content
With myRange
    .InsertAfter "This is a demonstration of SmallCaps."
    .Words(6).Font.SmallCaps = True
    .InsertParagraphAfter
    .InsertAfter "This is a demonstration of AllCaps."
    .Words(14).Font.AllCaps = True
End With
```

This example formats the entire selection as small capital letters if part of the selection is already formatted as small capital letters.

```
If Selection.Type = wdSelectionNormal Then
    mySel = Selection.Font.SmallCaps
    If mySel = wdUndefined Then Selection.Font.SmallCaps = True
Else
    MsgBox "You need to select some text."
End If
```



SmartCutPaste Property

-
True if Microsoft Word automatically adjusts the spacing between words and punctuation when cutting and pasting occurs. Read/write **Boolean**.

expression.**SmartCutPaste**

expression Required. An expression that returns an [Options](#) object.

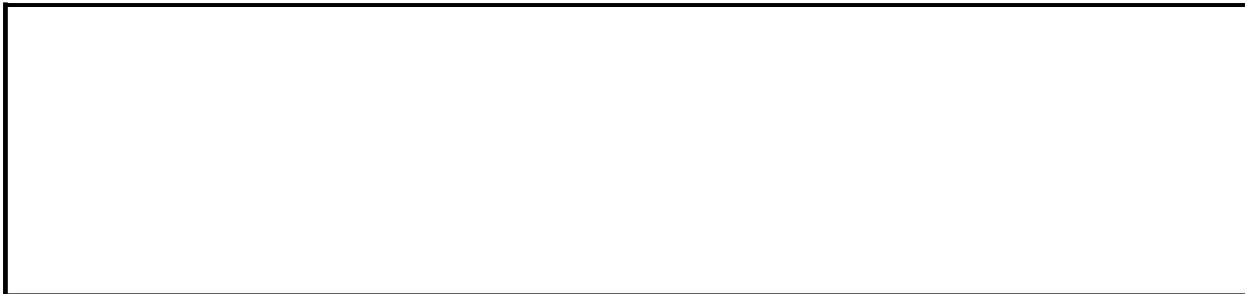
Example

This example sets Word to automatically adjust the spacing between words and punctuation when cutting and pasting occurs, and then it cuts and pastes some text in a newly created document. If the **SmartCutPaste** property were set to **False**, the second and third words would run together.

```
Options.SmartCutPaste = True
Set myDoc = Documents.Add
With myDoc
    .Content.InsertAfter("The brown quick fox")
    .Words(2).Cut
    .Characters(10).Paste
End With
```

This example returns the status of the **Smart cut and paste** option on the **Edit** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SmartCutPaste
```



SmartParaSelection Property

-
True for Microsoft Word to include the paragraph mark in a selection when selecting most or all of a paragraph. Read/write **Boolean**.

expression.**SmartParaSelection**

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

Example

This example disables smart paragraph selection.

```
Sub SetSmartParagraphSelection()  
    Options.SmartParaSelection = False  
End Sub
```



SmartTags Property

Returns a **SmartTags** object that represents a smart tag in a document.

expression.**SmartTags**

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

Example

This example adds custom properties to the first smart tag in the active document.

```
Sub NewSmartTagProperty()  
    ActiveDocument.SmartTags(1).Properties _  
        .Add Name:="President", Value:=True  
End Sub
```



SmartTagsAsXMLProps Property

-
True for Microsoft Word to create an XML header containing smart tag information when a document containing smart tags is saved as HTML. Read/write **Boolean**.

expression.**SmartTagsAsXMLProps**

expression Required. An expression that returns a [Document](#) object.

Example

This example enables saving smart tag information in an XML header if the active document is saved as HTML.

```
Sub SaveXMLForSmartTags()  
    ActiveDocument.SmartTagsAsXMLProps = True  
End Sub
```



SnapToGrid Property

-

Document object: **True** if AutoShapes or East Asian characters are automatically aligned with an invisible grid when they are drawn, moved, or resized in the specified document. Read/write **Boolean**.

Options object: **True** if AutoShapes or East Asian characters are automatically aligned with an invisible grid when they are drawn, moved, or resized in new documents. Read/write **Boolean**.

Remarks

You can temporarily override this setting by pressing ALT while drawing, moving, or resizing an AutoShape.

Example

This example sets Microsoft Word to automatically align East Asian characters with the invisible grid in the current document.

```
ActiveDocument.SnapToGrid = True
```

This example sets Word so that AutoShapes are automatically aligned with the invisible grid in a new document.

```
Options.SnapToGrid = True  
Documents.Add
```

This example returns the status of the **Snap to grid** option in the **Snap to Grid** dialog box (**Drawing** toolbar, **Draw** menu, **Grid** command).

```
Temp = Options.SnapToGrid
```



SnapToShapes Property

-

Document object: **True** if Microsoft Word automatically aligns AutoShapes or East Asian characters with invisible gridlines that go through the vertical and horizontal edges of other AutoShapes or East Asian characters in the specified document. Read/write **Boolean**.

Options object: **True** if Word automatically aligns AutoShapes or East Asian characters with invisible gridlines that go through the vertical and horizontal edges of other AutoShapes or East Asian characters in new documents. Read/write **Boolean**.

Remarks

This property creates additional invisible gridlines for each AutoShape. **SnapToShapes** works independently of the **SnapToGrid** property.

Example

This example sets Microsoft Word to automatically align East Asian characters with invisible gridlines that go through the vertical and horizontal edges of other East Asian characters in the current document.

```
ActiveDocument.SnapToShapes = True
```

This example sets Word to automatically align AutoShapes with invisible gridlines that go through the vertical and horizontal edges of other AutoShapes in a new document.

```
Options.SnapToShapes = True  
Documents.Add
```



↳ [Show All](#)

SortBy Property

Returns or sets the sorting criteria for the specified index. Read/write [WdIndexSortBy](#).

WdIndexSortBy can be one of these WdIndexSortBy constants.

wdIndexSortBySyllable Sort phonetically.

wdIndexSortByStroke Sort by the number of strokes in a character.

expression.SortBy

expression Required. An expression that returns an [Index](#) object.

Remarks

For more information on using Microsoft Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the first index in the current document to sort by the number of strokes.

```
ActiveDocument.Indexes(1).SortBy = _  
    wdIndexSortByStroke
```



SourceFullName Property

Returns or sets the path and name of the source file for the specified linked OLE object, picture, or field. Read/write **String**.

expression.**SourceFullName**

expression Required. An expression that returns a [LinkFormat](#) object.

Remarks

Using this property is equivalent to using in sequence the [SourcePath](#), [PathSeparator](#), and [SourceName](#) properties.

Example

This example sets MyExcel.xls as the source file for shape one on the active document and specifies that the OLE object be updated automatically.

```
With ActiveDocument.Shapes(1)
    If .Type = msoLinkedOLEObject Then
        With .LinkFormat
            .SourceFullName = "c:\my documents\myExcel.xls"
            .AutoUpdate = True
        End With
    End If
End With
```



SourceName Property

Returns the name of the source file for the specified linked OLE object, picture, or field. Read-only **String**.

expression.**SourceName**

expression Required. An expression that returns a [LinkFormat](#) object.

Remarks

This property doesn't return the path for the source file.

Example

This example returns the path and name of the source file for any shapes on the active document that are linked OLE objects.

```
For Each s In ActiveDocument.Shapes
    If s.Type = msoLinkedOLEObject Then
        MsgBox s.LinkFormat.SourcePath & "\" _
            & s.LinkFormat.SourceName
    End If
Next s
```



SourcePath Property

Returns the path of the source file for the specified linked OLE object, picture, or field. Read-only **String**.

expression.**SourcePath**

expression Required. An expression that returns a [LinkFormat](#) object.

Remarks

The path doesn't include a trailing character (for example, "C:\MSOffice"). Use the [PathSeparator](#) property to add the character that separates folders and drive letters. Use the [SourceName](#) property to return the file name without the path and use the [SourceFullName](#) property to return the path and file name together.

Example

This example returns the path and name of the source file for any shapes on the active document that are linked OLE objects.

```
For Each s In ActiveDocument.Shapes
    If s.Type = msoLinkedOLEObject Then
        MsgBox s.LinkFormat.SourcePath & "\" _
            & s.LinkFormat.SourceName
    End If
Next s
```



SpaceAfter Property

Returns or sets the amount of spacing (in points) after the specified paragraph or text column. Read/write **Single**.

expression.**SpaceAfter**

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

Example

This example sets the spacing after the first paragraph in the active document to 12 points.

```
ActiveDocument.Paragraphs(1).SpaceAfter = 12
```

This example sets the active document to three columns with a 0.5-inch space after the first column. The [InchesToPoints](#) method is used to convert inches to points.

```
With ActiveDocument.PageSetup.TextColumns  
    .SetCount NumColumns:=3  
    .LineBetween = False  
    .EvenlySpaced = True  
    .Item(1).SpaceAfter = InchesToPoints(0.5)  
End With
```



SpaceAfterAuto Property

-

True if Microsoft Word automatically sets the amount of spacing after the specified paragraphs. Returns **wdUndefined** if the **SpaceAfterAuto** property is set to **True** for only some of the specified paragraphs. Can be set to either **True** or **False**. Read/write **Long**.

Remarks

When you open an HTML document without cascading style sheets, Word automatically sets the **SpaceAfterAuto** property to **True** to render the paragraph spacing exactly as it would appear in a Web browser.

If **SpaceAfterAuto** is set to **True**, the [SpaceAfter](#) property is ignored.

Example

This example displays a report showing the **SpaceAfterAuto** settings for the active document.

```
Select Case ActiveDocument.Paragraphs.SpaceAfterAuto
  Case True
    x = "Spacing after paragraphs is handled " _
      & "automatically for all paragraphs."
  Case False
    x = "Spacing after paragraphs is handled " _
      & "manually for all paragraphs."
  Case wdUndefined
    x = "Spacing after paragraphs is handled " _
      & "automatically for some paragraphs, " _
      & "manually for some paragraphs."
End Select
```



SpaceBefore Property

Returns or sets the spacing (in points) before the specified paragraphs.
Read/write **Single**.

expression.**SpaceBefore**

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

Example

This example sets the spacing before the second paragraph in the active document to 12 points.

```
ActiveDocument.Paragraphs(2).SpaceBefore = 12
```



SpaceBeforeAuto Property

-

True if Microsoft Word automatically sets the amount of spacing before the specified paragraphs. Returns **wdUndefined** if the **SpaceBeforeAuto** property is set to **True** for only some of the specified paragraphs. Can be set to either **True** or **False**. Read/write **Long**.

Remarks

When you open an HTML document without cascading style sheets, Word automatically sets the **SpaceBeforeAuto** property to **True** to render the paragraph spacing exactly as it would appear in a Web browser.

If **SpaceBeforeAuto** is set to **True**, the [SpaceBefore](#) property is ignored.

Example

This example displays a report showing the **SpaceBeforeAuto** settings for the active document.

```
Select Case ActiveDocument.Paragraphs.SpaceBeforeAuto
  Case True
    x = "Spacing before paragraphs is handled " _
      & "automatically for all paragraphs."
  Case False
    x = "Spacing before paragraphs is handled " _
      & "manually for all paragraphs."
  Case wdUndefined
    x = "Spacing before paragraphs is handled " _
      & "automatically for some paragraphs, " _
      & "manually for some paragraphs."
End Select
```



SpaceBetweenColumns Property

Returns or sets the distance (in points) between text in adjacent columns of the specified row or rows. Read/write **Single**.

expression.**SpaceBetweenColumns**

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

Example

This example creates a 3x3 table in a new document and then sets the distance between columns in the first row to 0.5 inches.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Selection.Range, 3, 3)
myTable.Rows(1).SpaceBetweenColumns = InchesToPoints(0.5)
```

This example returns the distance (in points) between columns in the selected table rows.

```
If Selection.Information(wdWithInTable) = True Then
    MsgBox Selection.Rows.SpaceBetweenColumns
End If
```



Spacing Property

-

Returns or sets the spacing (in points) between characters (for the [Font](#) object), between the cells in a table (for the [Table](#) object), or between columns (for the [TextColumns](#) object). Read/write **Single**.

expression.**Spacing**

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

Remarks

After this property has been set for a **TextColumns** object, the [EvenlySpaced](#) property is set to **True**. To return or set the spacing for a single text column when [EvenlySpaced](#) is **False**, use the [SpaceAfter](#) property of the [TextColumn](#) object.

Example

This example demonstrates two different character spacings at the beginning of the active document.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
With myRange
    .InsertAfter "Demonstration of no character spacing."
    .InsertParagraphAfter
    .InsertAfter "Demonstration of character spacing (1.5pt)."
    .InsertParagraphAfter
End With
ActiveDocument.Paragraphs(2).Range.Font.Spacing = 1.5
```

This example sets the character spacing of the selected text to 2 points.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Spacing = 2
Else
    MsgBox "You need to select some text."
End If
```

This example sets the spacing between cells in the first table in the active document to nine points.

```
ActiveDocument.Tables(1).Spacing = 9
```

This example formats the active document to display text in two columns with 0.5 inch (36 points) spacing between the columns.

```
With ActiveDocument.PageSetup.TextColumns
    .SetCount NumColumns:=2
    .LineBetween = False
    .EvenlySpaced = True
    .Spacing = 36
End With
```



SpecialMode Property

True if Microsoft Word is in a special mode (for example, CopyText mode or MoveText mode). Read-only **Boolean**.

expression.**SpecialMode**

expression Required. An expression that returns an [Application](#) object.

Remarks

Word enters a special copy or move mode if you press F2 or SHIFT+F2 while text is selected.

Example

This example checks to see whether Word is in a special mode. If it is, ESC is activated before the current selection is cut and pasted.

```
If Application.SpecialMode = True Then SendKeys "ESC"  
With Selection  
    .Cut  
    .EndKey Unit:=wdStory  
    .Paste  
End With
```



SpellingChecked Property

-

True if spelling has been checked throughout the specified range or document.
False if all or some of the range or document hasn't been checked for spelling.
Read/write **Boolean**.

Remarks

To recheck the spelling in a range or document, set the **SpellingChecked** property to **False**.

To see whether the range or document contains spelling errors, use the **SpellingErrors** property.

Example

This example determines whether spelling in section one of the active document has been checked. If spelling hasn't been checked, the example starts a spelling check.

```
Set myRange = ActiveDocument.Sections(1).Range
isChecked = myRange.SpellingChecked
If isChecked = False Then
    myRange.CheckSpelling
Else
    MsgBox "The range has already been spell checked."
End If
```

This example sets the **SpellingChecked** property to **False** for MyDocument.doc, and then it runs another spelling check on the document.

```
Documents("MyDocument.doc").SpellingChecked = False
Documents("MyDocument.doc").CheckSpelling IgnoreUppercase:=False
```



↳ [Show All](#)

SpellingDictionaryType Property

Returns or sets the proofing tool type. Read/write [WdDictionaryType](#).

WdDictionaryType can be one of these WdDictionaryType constants.

wdGrammar

wdHangulHanjaConversion

wdHangulHanjaConversionCustom

wdHyphenation

wdSpelling

wdSpellingComplete

wdSpellingCustom

wdSpellingLegal

wdSpellingMedical

wdThesaurus

expression.**SpellingDictionaryType**

expression Required. An expression that returns a [Language](#) object.

Remarks

You can use this property to change the active spelling dictionary to one of the available add-on dictionaries that work with Word. For example, there are legal, medical, and complete spelling dictionaries you can use instead of the standard dictionary.

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.

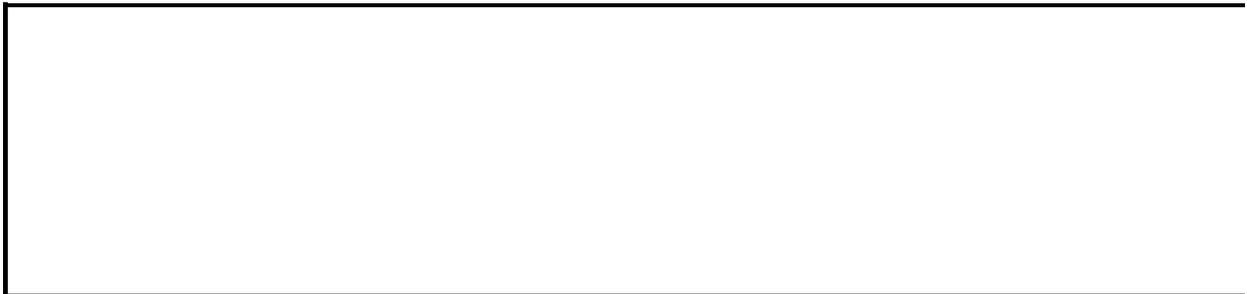
Example

This example returns the type of spelling dictionary used for U.S. English.

```
myType = Languages(wdEnglishUS).SpellingDictionaryType
```

This example makes the legal dictionary the active spelling dictionary.

```
Languages(wdEnglishUS).SpellingDictionaryType = wdSpellingLegal
```



SpellingErrors Property

Returns a [ProofreadingErrors](#) collection that represents the words identified as spelling errors in the specified document or range. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

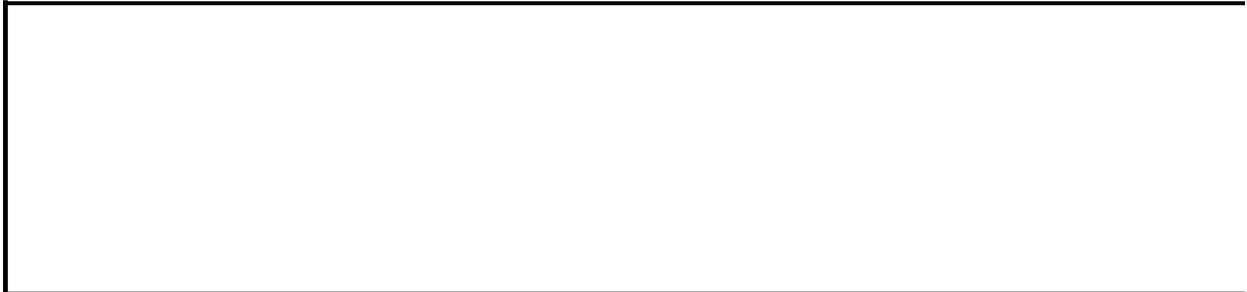
Example

This example checks the active document for spelling errors and displays the number of errors found.

```
myErr = ActiveDocument.SpellingErrors.Count
If myErr = 0 Then
    MsgBox "No spelling errors found."
Else
    MsgBox myErr & " spelling errors found."
End If
```

This example checks the specified range for spelling errors and displays each error found.

```
Set myErrors = ActiveDocument.Paragraphs(3).Range.SpellingErrors
If myErrors.Count = 0 Then
    MsgBox "No spelling errors found."
Else
    For Each myErr in myErrors
        MsgBox myErr.Text
    Next
End If
```



↳ [Show All](#)

SpellingErrorType Property

Returns the spelling error type. Read-only [WdSpellingErrorType](#).

WdSpellingErrorType can be one of these WdSpellingErrorType constants.

wdSpellingCapitalization

wdSpellingCorrect

wdSpellingNotInDictionary

expression.**SpellingErrorType**

expression Required. An expression that returns a [SpellingSuggestions](#) object.

Remarks

Use the [GetSpellingSuggestions](#) method to return a collection of words suggested as spelling replacements. If a word is misspelled, the [CheckSpelling](#) method returns **True**.

Example

If the first word in the active document isn't in the dictionary, this example displays "Unknown word" in the status bar.

```
Set suggs = ActiveDocument.Content.GetSpellingSuggestions
If suggs.SpellingErrorType = wdSpellingNotInDictionary Then
    StatusBar = "Unknown word"
End If
```



Split Property

-
True if the window is split into multiple panes. Read/write **Boolean**.

expression.**Split**

expression Required. An expression that returns a [Window](#) object.

Example

This example splits the active window into two equal-sized window panes.

```
ActiveDocument.ActiveWindow.Split = True
```

If the Document1 window is split, this example closes the active pane.

```
If Windows("Document1").Split = True Then  
    Windows("Document1").ActivePane.Close  
End If
```



SplitSpecial Property

Returns or sets the active window pane. Read/write [WdSpecialPane](#).

Can be one of the following **WdSpecialPane** constants:

wdPaneComments

wdPaneFirstPageFooter

wdCurrentPageFooter

wdPaneFirstPageHeader

wdPaneCurrentPageHeader

wdPaneFootnoteContinuationNotice

wdPaneEndnoteContinuationNotice

wdPaneFootnoteContinuationSepar

wdPaneEndnoteContinuationSeparator

wdPaneFootnotes

wdPaneEndnotes

wdPaneFootnoteSeparator

wdPaneEndnoteSeparator

wdPaneNone

wdPaneEvenPagesFooter

wdPanePrimaryFooter

wdPaneEvenPagesHeader

wdPanePrimaryHeader

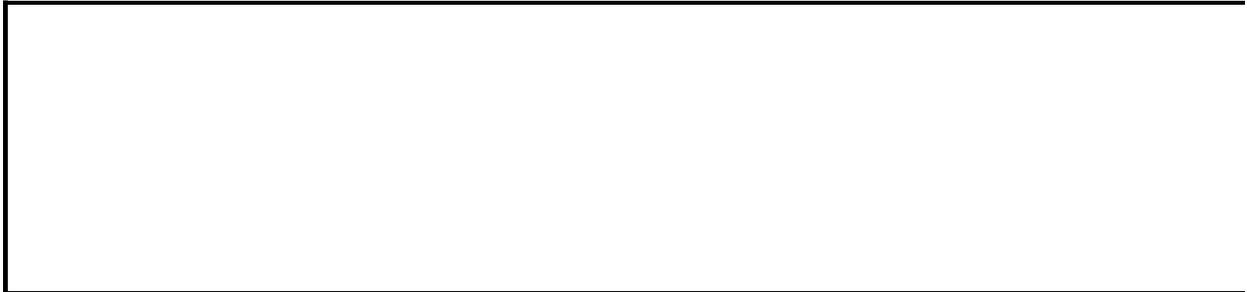
Example

This example displays the primary footer in a separate pane in the active window.

```
ActiveDocument.ActiveWindow.View.SplitSpecial = wdPanePrimaryFooter
```

This example adds a footnote to the active document and displays all the footnotes in a separate pane in the active window.

```
ActiveDocument.Footnotes.Add Range:=Selection.Range, _  
    Text:="Footnote text"  
With ActiveDocument.ActiveWindow.View  
    .Type = wdNormalView  
    .SplitSpecial = wdPaneFootnotes  
End With
```



SplitVertical Property

Returns or sets the vertical split percentage for the specified window. Read/write **Long**.

expression.**SplitVertical**

expression Required. An expression that returns a [Window](#) object.

Remarks

To remove the split, set this property to zero (0) or set the [Split](#) property to **False**.

Example

This example splits the active window so that the top pane occupies 70 percent of the window.

```
ActiveDocument.ActiveWindow.SplitVertical = 70
```

This example splits the window for Document1 in half vertically.

```
Windows("Document1").SplitVertical = 50
```



Start Property

Returns or sets the starting character position of a selection, range, or bookmark.
Read/write **Long**.

Note 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.

Remarks

Selection, **Range**, and **Bookmark** objects have starting and ending character positions. The starting position refers to the character position closest to the beginning of the story.

This property returns the starting character position relative to the beginning of the story. The main text story (**wdMainTextStory**) begins with character position 0 (zero). You can change the size of a selection, range, or bookmark by setting this property.

Example

This example returns the starting position of the second paragraph and the ending position of the fourth paragraph in the active document. The character positions are used to create the range myRange.

```
pos = ActiveDocument.Paragraphs(2).Range.Start  
pos2 = ActiveDocument.Paragraphs(4).Range.End  
Set myRange = ActiveDocument.Range(Start:=pos, End:=pos2)
```

This example determines the length of the selection by comparing the starting and ending character positions.

```
SelLength = Selection.End - Selection.Start
```

This example moves the starting position of myRange one character to the right (this reduces the size of the range by one character).

```
Set myRange = Selection.Range  
myRange.SetRange Start:=myRange.Start + 1, End:=myRange.End
```



StartAt Property

Returns or sets the starting number for the specified [ListLevel](#) object.
Read/write **Long**.

expression.**StartAt**

expression Required. An expression that returns a [ListLevel](#) object.

Example

This example sets the number style and starting number for the third outline-numbered list template. Because the style uses uppercase letters and the starting number is 4, the first letter is D.

```
Set mylev = ListGalleries(wdOutlineNumberGallery) _  
    .ListTemplates(3).ListLevels(1)  
With mylev  
    .NumberStyle = wdListNumberStyleUppercaseLetter  
    .StartAt = 4  
End With
```



StartingNumber Property

Returns or sets the starting note number, line number, or page number.
Read/write **Long**.

expression.**StartingNumber**

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

Remarks

You must be in print layout view to see line numbering.

When applied to page numbers, this property returns or sets the beginning page number for the specified [HeaderFooter](#) object. This number may or may not be visible on the first page, depending on the setting of the [ShowFirstPageNumber](#) property. The [RestartNumberingAtSection](#) property, if set to **False**, will override the **StartingNumber** property so that page numbering can continue from the previous section.

Example

This example creates a new document, sets the starting number for footnotes to 10, and then adds a footnote at the insertion point.

```
Set myDoc = Documents.Add
With myDoc.Footnotes
    .StartingNumber = 10
    .Add Range:=Selection.Range, Text:="Text for a footnote"
End With
```

This example enables line numbering for the active document. The starting number is set to 5, every fifth line number is shown, and the numbering starts over at the beginning of each section in the document.

```
With ActiveDocument.PageSetup.LineNumbering
    .Active = True
    .StartingNumber = 5
    .CountBy = 5
    .RestartMode = wdRestartSection
End With
```

This example sets properties for page numbers, and then it adds page numbers to the header of the active document.

```
With ActiveDocument.Sections(1) _
    .Headers(wdHeaderFooterPrimary).PageNumbers
    .NumberStyle = wdPageNumberStyleArabic
    .IncludeChapterNumber = False
    .RestartNumberingAtSection = True
    .StartingNumber = 5
    .Add PageNumberAlignment:=wdAlignPageNumberCenter, _
        FirstPage:=True
End With
```



StartIsActive Property

True if the beginning of the selection is active. If the selection is not collapsed to an insertion point, either the beginning or the end of the selection is active. The active end of the selection moves when you call the following methods: [EndKey](#), [Extend](#) (with the *Characters* argument), [HomeKey](#), [MoveDown](#), [MoveLeft](#), [MoveRight](#), and [MoveUp](#). Read/write **Boolean**.

expression.**StartIsActive**

expression Required. An expression that returns a [Selection](#) object.

Remarks

This property is equivalent to using the [Flags](#) property with the **wdSelStartActive** constant. However, using the **Flags** property requires binary operations, which are more complicated than using the **StartIsActive** property.

Example

This example extends the current selection through the next two words. To make sure that any currently selected text stays selected during the extension, the end of the selection is made active first. (For example, if the first three words of this paragraph were selected but the start of the selection were active, the **MoveRight** method call would simply deselect the first two words.)

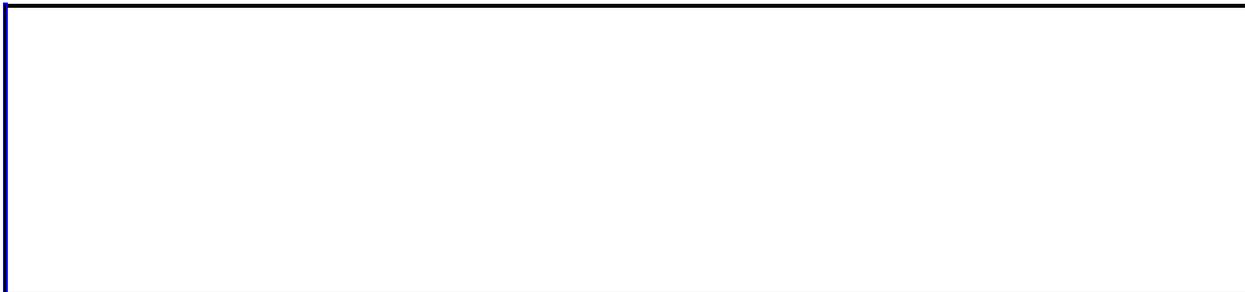
```
With Selection
    .StartIsActive = False
    .MoveRight Unit:=wdWord, Count:=2, Extend:=wdExtend
End With
```

Here's the same example using the **Flags** property. This solution is problematic because you can only deactivate a **Flags** property setting by overwriting it with an unrelated value.

```
With Selection
    If (.Flags And wdSelStartActive) = wdSelStartActive Then _
        .Flags = wdSelReplace
        .MoveRight Unit:=wdWord, Count:=2, Extend:=wdExtend
End With
```

Here's the same example using the [MoveEnd](#) method, which eliminates the need to check which end of the selection is active.

```
With Selection
    .MoveEnd Unit:=wdWord, Count:=2
End With
```



StartupPath Property

Returns or sets the complete path of the startup folder, excluding the final separator. Read/write **String**.

expression.**StartupPath**

expression Required. An expression that returns an [Application](#) object.

Remarks

Templates and add-ins located in the Startup folder are automatically loaded when you start Word.

Example

This example displays the complete path of the Startup folder.

```
MsgBox Application.StartupPath
```

This example enables the user to change the path of the Startup folder.

```
x = MsgBox("Do you want to change the startup path?", vbYesNo, _  
    "Current path = " & Application.StartupPath)  
If x = vbYes Then  
    newStartup = InputBox("Type a startup path")  
    Application.StartupPath = newStartup  
End If
```



↳ [Show All](#)

State Property

-
Returns the current state of a mail merge operation. Read-only
[WdMailMergeState](#).

WdMailMergeState can be one of these WdMailMergeState constants.

wdDataSource

wdMainAndDataSource

wdMainAndHeader

wdMainAndSourceAndHeader

wdMainDocumentOnly

wdNormalDocument

expression.State

expression Required. An expression that returns a [MailMerge](#) object.

Example

This example executes a mail merge if the active document is a main document with an attached data source.

```
Set myMerge = ActiveDocument.MailMerge  
If myMerge.State = wdMainAndDataSource Then myMerge.Execute
```



↳ [Show All](#)

Status Property

Returns the routing status of the specified routing slip. Read-only [WdRoutingSlipStatus](#).

WdRoutingSlipStatus can be one of these WdRoutingSlipStatus constants.

wdNotYetRouted

wdRouteComplete

wdRouteInProgress

expression.Status

expression Required. An expression that returns a [RoutingSlip](#) object.

Example

If the active document has a routing slip attached to it, this example displays a message indicating the routing status.

```
If ActiveDocument.HasRoutingSlip = True Then
    Select Case ActiveDocument.RoutingSlip.Status
        Case wdNotYetRouted
            MsgBox "The document hasn't been routed yet."
        Case wdRouteInProgress
            MsgBox "Routing is in progress."
        Case wdRouteComplete
            MsgBox "Routing is complete."
    End Select
End If
```

This example resets the routing slip for Sales.doc if the routing is complete.

```
With Documents("Sales.doc").RoutingSlip
    If .Status = wdRouteComplete Then
        .Reset
    Else
        MsgBox "Cannot reset routing; not yet complete."
    End If
End With
```



StatusBar Property

-
Displays the specified text in the status bar. Write-only **String**.

expression.**StatusBar**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays a message in the status bar.

```
StatusBar = "Please wait..."
```

This example displays in the status bar the name of the template attached to the active document.

```
aName = ActiveDocument.AttachedTemplate.Name  
StatusBar = aName & " template is attached to the active document"
```



StatusText Property

Returns or sets the text that's displayed in the status bar when a form field has the focus. If the [OwnStatus](#) property is set to **True**, the **StatusText** property specifies the status bar text. If the **OwnStatus** property is set to **False**, the **StatusText** property specifies the name of an AutoText entry that contains status bar text for the form field. Read/write **String**.

expression.**StatusText**

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

Example

This example sets the status bar help text for the form field named "Age."

```
With ActiveDocument.FormFields("Age")  
    .OwnStatus = True  
    .StatusText = "Type your current age."  
End With
```

A large empty rectangular box with a black border, representing a form field. It is positioned below the code block and occupies a significant portion of the lower half of the page.

StoreRSIDOnSave Property

-

True for Microsoft Word to assign a random number to changes in a document, each time a document is saved, to facilitate comparing and merging documents. Word stores the random numbers in a table and updates the table after each save. Read/write **Boolean**.

expression.**StoreRSIDOnSave**

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

Remarks

The default for the **StoreRSIDOnSave** property is **True**. However, RSID information is not saved for HTML documents.

Use the [RemovePersonalInformation](#) property if you want to remove information related to authors and reviewers of a document.

Example

This example turns off storing a random number when saving documents.

```
Sub SaveRandomNumber()  
    Application.Options.StoreRSIDOnSave = False  
End Sub
```



↳ [Show All](#)

StoryLength Property

-
Returns the number of characters in the [story](#) that contains the specified range or selection. Read-only **Long**.

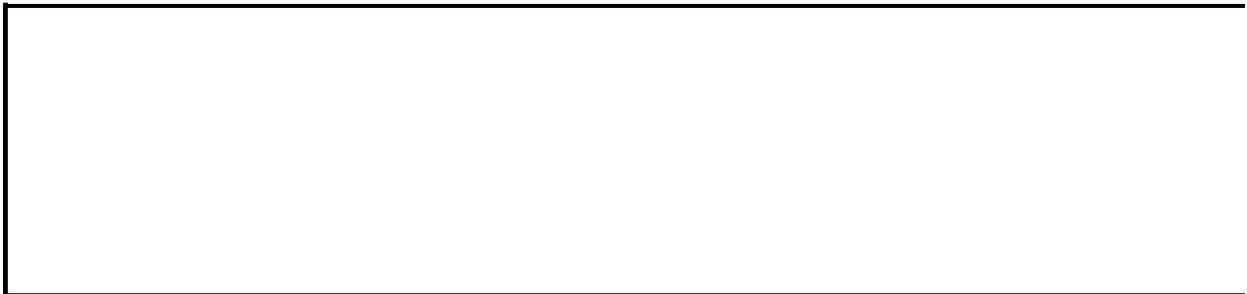
Example

This example determines whether the header in the active document is empty. If the header story isn't empty, a message box displays the contents of the header. If the document header is empty, **StoryLength** returns 1 for the final paragraph mark.

```
Set myRange = ActiveDocument.Sections(1) _  
    .Headers(wdHeaderFooterPrimary).Range  
If myRange.StoryLength > 1 Then MsgBox myRange.Text
```

This example closes the document without saving changes if it's empty.

```
If ActiveDocument.Content.StoryLength = 1 Then _  
    ActiveDocument.Close SaveChanges:=wdDoNotSaveChanges
```



StoryRanges Property

Returns a [StoryRanges](#) collection that represents all the stories in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

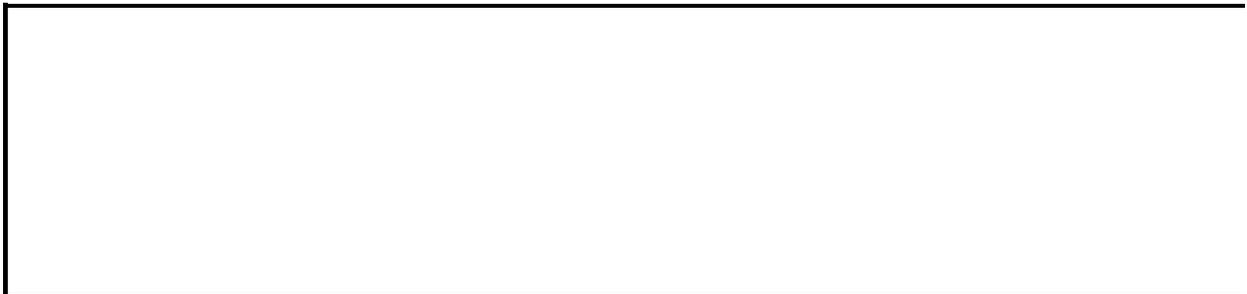
Example

This example steps through the **StoryRanges** collection to determine whether **wdPrimaryFooterStory** is part of the **StoryRanges** collection.

```
For Each aStory In ActiveDocument.StoryRanges
    If aStory.StoryType = wdEvenPagesFooterStory Then
        MsgBox "Document includes an even page footer"
    End If
Next aStory
```

This example adds text to the primary header story and then displays the text.

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary).Range _
    .Text = "Header text"
MsgBox ActiveDocument.StoryRanges(wdPrimaryHeaderStory).Text
```



↳ [Show All](#)

StoryType Property

Returns the [story](#) type for the specified range, selection, or bookmark. Read-only [WdStoryType](#).

WdStoryType can be one of these WdStoryType constants.

wdCommentsStory

wdEndnotesStory

wdEvenPagesFooterStory

wdEvenPagesHeaderStory

wdFirstPageFooterStory

wdFirstPageHeaderStory

wdFootnotesStory

wdMainTextStory

wdPrimaryFooterStory

wdPrimaryHeaderStory

wdTextFrameStory

expression.**StoryType**

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

Example

This example returns the story type of the selection.

```
story = Selection.StoryType
```

This example closes the footnote pane if the selection is contained in the footnote story.

```
ActiveDocument.ActiveWindow.View.Type = wdNormalView  
If Selection.StoryType = wdFootnotesStory Then _  
    ActiveDocument.ActiveWindow.ActivePane.Close
```

This example selects the bookmark named "temp" if the bookmark is contained in the main story of the active document.

```
If ActiveDocument.Bookmarks.Exists("temp") = True Then  
    Set myBookmark = ActiveDocument.Bookmarks("temp")  
    If myBookmark.StoryType = wdMainTextStory _  
        Then myBookmark.Select  
End If
```



StrictFinalYaa Property

-
True if the spelling checker uses spelling rules regarding Arabic words ending with the letter *yaa*. Read/write **Boolean**.

expression.**StrictFinalYaa**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the spelling checker to use spelling rules regarding Arabic words ending with the letter *yaa*.

Options.**StrictFinalYaa** = True



StrictInitialAlefHamza Property

-
True if the spelling checker uses spelling rules regarding Arabic words beginning with an *alef hamza*. Read/write **Boolean**.

expression.**StrictInitialAlefHamza**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the spelling checker to use spelling rules regarding Arabic words beginning with an *alef hamza*.

Options.**StrictInitialAlefHamza** = True



StrikeThrough Property

-
True if the font is formatted as strikethrough text. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**StrikeThrough**

expression Required. An expression that returns a [Font](#) object.

Remarks

To set or return double strikethrough formatting, use the [DoubleStrikeThrough](#) property.

Example

This example applies strikethrough formatting to the first three words in the active document.

```
Set myDoc = ActiveDocument
Set myRange = myDoc.Range(Start:=myDoc.Words(1).Start, _
    End:=myDoc.Words(3).End)
myRange.Font.StrikeThrough = True
```

This example applies strikethrough formatting to the selected text.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.StrikeThrough = True
Else
    MsgBox "You need to select some text."
End If
```



↳ [Show All](#)

Style Property

▶ [Style property as it applies to the **LineFormat** object.](#)

Returns or sets the line format style. Read/write [MsoLineStyle](#).

MsoLineStyle can be one of these MsoLineStyle constants.

msoLineSingle

msoLineThickBetweenThin

msoLineThinThick

msoLineStyleMixed

msoLineThickThin

msoLineThinThin

expression.**Style**

expression Required. An expression that returns a [LineFormat](#) object.

▶ [Style property as it applies to the **EmailAuthor** and **Revision** objects.](#)

Returns a [Style](#) object that represents the style associated with the current e-mail author for unsent replies, forwards, or new e-mail messages.

expression.**Style**

expression Required. An expression that returns one of the above objects.

▶ [Style property as it applies to the **Find**, **HeadingStyle**, **Paragraph**, **ParagraphFormat**, **Paragraphs**, **Range**, **Replacement**, **Selection**, and **Table** objects.](#)

Returns or sets the style for the specified object. To set this property, specify the local name of the style, an integer, a **WdBuiltinStyle** constant, or an object that represents the style. For a list of valid constants, consult the Microsoft Visual

Basic Object Browser. Read/write **Variant**.

expression.**Style**

expression Required. An expression that returns one of the above objects.

Remarks

When you return the style for a range that includes more than one style, only the first character or paragraph style is returned.

Example

▶ [As it applies to the **EmailAuthor** object.](#)

This example returns the style associated with the current author for unsent replies, forwards, or new e-mail messages and displays the name of the font associated with this style.

```
Set MyEmailStyle = _  
    ActiveDocument.Email.CurrentEmailAuthor.Style  
Msgbox MyEmailStyle.Font.Name
```

▶ [As it applies to the **Paragraph** object.](#)

This example displays the style for each paragraph in the active document.

```
For Each para in ActiveDocument.Paragraphs  
    MsgBox para.Style  
Next para
```

This example sets alternating styles of Heading 3 and Normal for all the paragraphs in the active document.

```
For i = 1 To ActiveDocument.Paragraphs.Count  
    If i Mod 2 = 0 Then  
        ActiveDocument.Paragraphs(i).Style = wdStyleNormal  
    Else: ActiveDocument.Paragraphs(i).Style = wdStyleHeading3  
    End If  
Next i
```

▶ [As it applies to the **Range** object.](#)

This example displays the style for each character in the selection. Each element of the **Characters** collection is a **Range** object.

```
For each c in Selection.Characters  
    MsgBox c.Style  
Next c
```



StyleAreaWidth Property

Returns or sets the width of the style area in points. Read/write **Single**.

expression.**StyleAreaWidth**

expression Required. An expression that returns a [Window](#) object.

Remarks

When the **StyleAreaWidth** property is greater than 0 (zero), style names are displayed to the left of the text. The style area isn't visible in print layout or Web layout view.

Example

This example switches the active window to normal view and sets the width of the style area to 1 inch.

```
With ActiveDocument.ActiveWindow  
    .View.Type = wdNormalView  
    .StyleAreaWidth = InchesToPoints(1)  
End With
```



StyleName Property

Returns the name of the style applied to the specified AutoText entry. Read-only **String**.

expression.**StyleName**

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

Example

This example creates an AutoText entry and then displays the style name of the entry.

```
Set myentry = NormalTemplate.AutoTextEntries.Add(Name:="rsvp", _  
    Range:=Selection.Range)  
MsgBox myentry.StyleName
```



Styles Property

Returns a [Styles](#) collection for the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example applies the Heading 1 style to each paragraph in the active document that begins with the word "Chapter."

```
For Each para In ActiveDocument.Paragraphs
    If para.Range.Words(1).Text = "Chapter " Then
        para.Style = ActiveDocument.Styles(wdStyleHeading1)
    End If
Next para
```

This example opens the template attached to the active document and modifies the Heading 1 style. The template is saved, and all styles in the active document are updated.

```
Set tempDoc = ActiveDocument.AttachedTemplate.OpenAsDocument
With tempDoc.Styles(wdStyleHeading1).Font
    .Name = "Arial"
    .Size = 16
End With
tempDoc.Close SaveChanges:=wdSaveChanges
ActiveDocument.UpdateStyles
```



StyleSheets Property

Returns a [StyleSheets](#) object that represents the Web style sheets attached to a document.

expression.**StyleSheets**

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

Example

This example adds a style sheet to the active document and places it highest in the list of style sheets attached to the document. This example assumes that you have a style sheet document named "Website.css" located on your C: drive.

```
Sub Styshts()  
    ThisDocument.StyleSheets.Add _  
        FileName:="c:\WebSite.css", _  
        Precedence:=wdStyleSheetPrecedenceHighest  
End Sub
```



SubAddress Property

Returns or sets a named location in the destination of the specified hyperlink.
Read/write **String**.

expression.**SubAddress**

expression Required. An expression that returns a [Hyperlink](#) object.

Remarks

The named location can be a bookmark in a Microsoft Word document, a named cell or cell reference in a Microsoft Excel worksheet, a named object in a Microsoft Access database, or a slide number in a Microsoft PowerPoint presentation.

Example

This example displays the subaddress of the selected hyperlink.

```
If Selection.Range.Hyperlinks.Count >= 1 Then
    MsgBox Selection.Range.Hyperlinks(1).SubAddress
End If
```

This example adds a hyperlink to the selection in the active document, sets the hyperlink destination and subaddress, and then displays them in a message box.

```
Set SCut = ActiveDocument.Hyperlinks.Add( _
    Anchor:= Selection.Range, _
    Address:="C:\My Documents\Other.doc", SubAddress:= "temp")
MsgBox "The hyperlink goes to " & SCut.SubAddress
```



Subdocuments Property

Returns a [Subdocuments](#) collection that represents all the subdocuments in the specified range or document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the number of subdocuments embedded in the active document.

```
MsgBox ActiveDocument.Subdocuments.Count
```

This example displays the path and file name of each subdocument in the active document.

```
For Each subdoc In ActiveDocument.Subdocuments  
    If subdoc.HasFile = True Then  
        MsgBox subdoc.Path & Application.PathSeparator _  
            & subdoc.Name  
    Else  
        MsgBox "This subdocument has not been saved."  
    End If  
Next subdoc
```



Subject Property

-

Returns or sets the subject text of mail messages used to route a document (for the [RoutingSlip](#) object) or the subject text of a letter created by the Letter Wizard (for the [LetterContent](#) object). Read/write **String**.

expression.**Subject**

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

Example

This example sets the subject and message text for the routing slip associated with Month End.doc.

```
If Documents("Month End.doc").HasRoutingSlip = True Then
    With Documents("Month End.doc").RoutingSlip
        .Subject = "End of month report"
        .Message = "I need your response on this."
    End With
End If
```

This example displays the subject of a letter created by the Letter Wizard, unless the subject is an empty string.

```
If ActiveDocument.GetLetterContent.Subject <> "" Then
    MsgBox ActiveDocument.GetLetterContent.Subject
End If
```



Subscript Property

-
True if the font is formatted as subscript. Returns **True**, **False** or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**.
Read/write **Long**.

expression.**Subscript**

expression Required. An expression that returns a [Font](#) object.

Remarks

Setting the **Subscript** property to **True** sets the [Superscript](#) property to **False**, and vice versa.

Example

This example inserts text at the beginning of the active document and formats the tenth character as subscript.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
myRange.InsertAfter "Water = H20"
myRange.Characters(10).Font.Subscript = True
```

This example checks the selected text for subscript formatting.

```
If Selection.Type = wdSelectionNormal Then
    mySel = Selection.Font.Subscript
    If mySel = wdUndefined Or mySel = True Then
        MsgBox "Subscript text exists in the selection."
    Else
        MsgBox "No subscript text in the selection."
    End If
Else
    MsgBox "You need to select some text."
End If
```



SuggestFromMainDictionaryOnly Property

True if Microsoft Word draws spelling suggestions from the main dictionary only. **False** if it draws spelling suggestions from the main dictionary and any custom dictionaries that have been added. Read/write **Boolean**.

expression.**SuggestFromMainDictionaryOnly**

expression Required. An expression that returns an [Options](#) object.

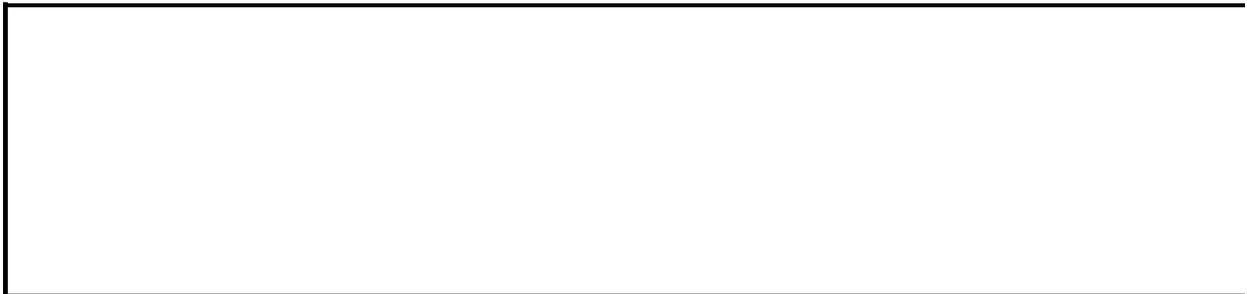
Example

This example sets Word to suggest words from the main dictionary only, and then it checks spelling in the active document.

```
Options.SuggestFromMainDictionaryOnly = True  
ActiveDocument.CheckSpelling
```

This example returns the current status of the **Suggest from main dictionary only** option on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SuggestFromMainDictionaryOnly
```



SuggestSpellingCorrections Property

True if Microsoft Word always suggests alternative spellings for each misspelled word when checking spelling. Read/write **Boolean**.

expression.**SuggestSpellingCorrections**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to always suggest alternative spellings for misspelled words, and then it checks spelling in the active document.

```
Options.SuggestSpellingCorrections = True  
ActiveDocument.CheckSpelling
```

This example returns the current status of the **Always suggest corrections** option on the **Spelling & Grammar** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.SuggestSpellingCorrections
```



SummaryLength Property

Returns or sets the length of the summary as a percentage of the document length. The larger the number, the more detail that's included in the summary. Read/write **Long**.

Note This property takes effect immediately if the **AutoSummarize** toolbar is displayed; otherwise, it takes effect the next time the **AutoSummarize** method or the **SummaryViewMode** property is applied to the document.

Example

This example highlights the key points in the active document. The level of detail is set to 50 percent.

```
With ActiveDocument
    .AutoSummarize Mode:=wdSummaryModeHighlight
    .SummaryLength = 50
End With
```

This example displays the summary and sets the level of detail to 55 percent.

```
With ActiveDocument
    .ShowSummary = True
    .SummaryLength = 55
End With
```



SummaryViewMode Property

Returns or sets the way a summary is displayed. This property corresponds to **Type of summary** in the **AutoSummarize** dialog box (**Tools** menu). Read/write [WdSummaryMode](#).

Can be one of the following **WdSummaryMode** constants.

Constant	Description
wdSummaryModeHighlight	Highlights the key points in the specified document and displays the AutoSummarize toolbar.
wdSummaryModeInsert	Inserts a summary at the beginning of the specified document.
wdSummaryModeCreateNew	Creates a new document and inserts the specified summary.
wdSummaryModeHideAllButSummary	Hides everything except the specified summary and displays the AutoSummarize toolbar.

Example

This example hides everything in the active document except the summary text.

```
With ActiveDocument
    .SummaryViewMode = wdSummaryModeHideAllButSummary
    .SummaryLength = 60
    .ShowSummary = True
End With
```



Superscript Property

-
True if the font is formatted as superscript. Returns **True**, **False**, or **wdUndefined** (a mixture of **True** and **False**). Can be set to **True**, **False**, or **wdToggle**. Read/write **Long**.

expression.**Superscript**

expression Required. An expression that returns a [Font](#) object.

Remarks

Setting the **Superscript** property to **True** sets the [Subscript](#) property to **False**, and vice versa.

Example

This example inserts text at the beginning of the active document and formats two characters in the fourth word as superscript.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
myRange.InsertAfter "Superscript in the 4th word."
ActiveDocument.Range(Start:=20, End:=22).Font.Superscript = True
```

This example formats the selected text as superscript.

```
If Selection.Type = wdSelectionNormal Then
    Selection.Font.Superscript = True
Else
    MsgBox "You need to select some text."
End If
```



SuppressBlankLines Property

True if blank lines are suppressed when mail merge fields in a mail merge main document are empty. Read/write **Boolean**.

expression.**SuppressBlankLines**

expression Required. An expression that returns a [MailMerge](#) object.

Example

This example opens Main.doc and executes the mail merge operation. When merge fields are empty, blank lines are suppressed in the merge document.

```
Set myDoc = Documents.Open(FileName:="C:\My Documents\Main.doc")
With myDoc.MailMerge
    .SuppressBlankLines = True
    .Destination = wdSendToPrinter
    .Execute
End With
```



SuppressEndnotes Property

-
True if endnotes are printed at the end of the next section that doesn't suppress endnotes. Suppressed endnotes are printed before the endnotes in that section.
Read/write **Long**.

expression.**SuppressEndnotes**

expression Required. An expression that returns a [PageSetup](#) object.

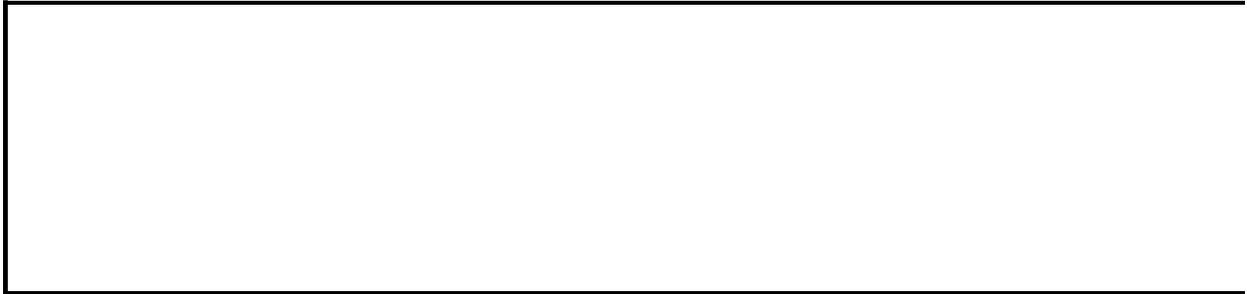
Remarks

This property takes effect only if the [Location](#) property is set to **wdEndOfSection**.

Example

This example suppresses endnotes in the first section of the active document.

```
If ActiveDocument.Endnotes.Location = wdEndOfSection Then  
    ActiveDocument.Sections(1).PageSetup.SuppressEndnotes = True  
End If
```



SurroundFooter Property

True if a page border encompasses the document footer. Read/write **Boolean**.

expression.**SurroundFooter**

expression Required. An expression that returns a [Borders](#) collection object.

Example

This example formats the page border in section one of the active document so that it encompasses the header and footer on each page in the section.

```
With ActiveDocument.Sections(1).Borders
    .SurroundFooter = True
    .SurroundHeader = True
End With
```

This example adds a graphical page border around each page in section one. The page border doesn't encompass the header and footer areas.

```
With ActiveDocument.Sections(1)
    .Borders.SurroundFooter = False
    .Borders.SurroundHeader = False
    For Each aBord In .Borders
        aBord.ArtStyle = wdArtPeople
        aBord.ArtWidth = 15
    Next aBord
End With
```



SurroundHeader Property

True if a page border encompasses the document header. Read/write **Boolean**.

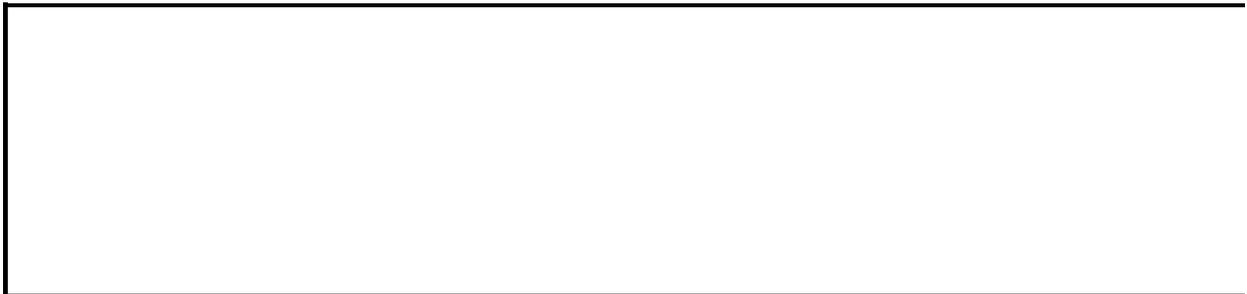
expression.**SurroundHeader**

expression Required. An expression that returns a [Borders](#) collection object.

Example

This example formats the page border in section one of the active document to exclude the header and footer areas on each page.

```
With ActiveDocument.Sections(1).Borders  
    .SurroundFooter = False  
    .SurroundHeader = False  
End With
```



↳ [Show All](#)

SynonymInfo Property

▶ [SynonymInfo property as it applies to the **Range** object.](#)

Returns a **SynonymInfo** object that contains information from the thesaurus on synonyms, antonyms, or related words and expressions for the specified word or phrase.

expression.**SynonymInfo**

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

▶ [SynonymInfo property as it applies to the **Application** and **Global** objects.](#)

Returns a **SynonymInfo** object that contains information from the thesaurus on synonyms, antonyms, or related words and expressions for the specified word or phrase.

expression.**SynonymInfo**(*Word*, *LanguageID*)

expression Required. An expression that returns one of the above objects.

Word Required **String**. The specified word or phrase.

LanguageID Optional **Variant**. The language used for the thesaurus. Can be one of the **WdLanguageID** constants (although some of the constants listed may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed).

WdLanguageID can be one of these WdLanguageID constants.

wdAfrikaans

wdAlbanian

wdArabic

wdArabicAlgeria

wdArabicBahrain

wdArabicEgypt
wdArabicIraq
wdArabicJordan
wdArabicKuwait
wdArabicLebanon
wdArabicLibya
wdArabicMorocco
wdArabicOman
wdArabicQatar
wdArabicSyria
wdArabicTunisia
wdArabicUAE
wdArabicYemen
wdArmenian
wdAssamese
wdAzeriCyrillic
wdAzeriLatin
wdBasque
wdBelgianDutch
wdBelgianFrench
wdBengali
wdBosniaHerzegovina
wdBrazilianPortuguese
wdBulgarian
wdBurmese
wdByelorussian
wdCatalan
wdChineseHongKong
wdChineseMacao
wdChineseSingapore
wdCroatian
wdCzech
wdDanish

wdDutch
wdEnglishAUS
wdEnglishBelize
wdEnglishCanadian
wdEnglishCaribbean
wdEnglishIreland
wdEnglishJamaica
wdEnglishNewZealand
wdEnglishPhilippines
wdEnglishSouthAfrica
wdEnglishTrinidad
wdEnglishUK
wdEnglishUS
wdEnglishZimbabwe
wdEstonian
wdFaeroese
wdFarsi
wdFinnish
wdFrench
wdFrenchCameroon
wdFrenchCanadian
wdFrenchCotedIvoire
wdFrenchLuxembourg
wdFrenchMali
wdFrenchMonaco
wdFrenchReunion
wdFrenchSenegal
wdFrenchWestIndies
wdFrenchZaire
wdFrisianNetherlands
wdGaelicIreland
wdGaelicScotland
wdGalician

wdGeorgian
wdGerman
wdGermanAustria
wdGermanLiechtenstein
wdGermanLuxembourg
wdGreek
wdGujarati
wdHebrew
wdHindi
wdHungarian
wdIcelandic
wdIndonesian
wdItalian
wdJapanese
wdKannada
wdKashmiri
wdKazakh
wdKhmer
wdKirghiz
wdKonkani
wdKorean
wdLanguageNone
wdLao
wdLatvian
wdLithuanian
wdLithuanianClassic
wdMacedonian
wdMalayalam
wdMalayBruneiDarussalam
wdMalaysian
wdMaltese
wdManipuri
wdMarathi

wdMexicanSpanish
wdMongolian
wdNepali
wdNoProofing
wdNorwegianBokmol
wdNorwegianNynorsk
wdOriya
wdPolish
wdPortuguese
wdPunjabi
wdRhaetoRomanic
wdRomanian
wdRomanianMoldova
wdRussian
wdRussianMoldova
wdSamiLappish
wdSanskrit
wdSerbianCyrillic
wdSerbianLatin
wdSesotho
wdSimplifiedChinese
wdSindhi
wdSlovak
wdSlovenian
wdSorbian
wdSpanish
wdSpanishArgentina
wdSpanishBolivia
wdSpanishChile
wdSpanishColombia
wdSpanishCostaRica
wdSpanishDominicanRepublic
wdSpanishEcuador

wdSpanishElSalvador
wdSpanishGuatemala
wdSpanishHonduras
wdSpanishModernSort
wdSpanishNicaragua
wdSpanishPanama
wdSpanishParaguay
wdSpanishPeru
wdSpanishPuertoRico
wdSpanishUruguay
wdSpanishVenezuela
wdSutu
wdSwahili
wdSwedish
wdSwedishFinland
wdSwissFrench
wdSwissGerman
wdSwissItalian
wdTajik
wdTamil
wdTatar
wdTelugu
wdThai
wdTibetan
wdTraditionalChinese
wdTsonga
wdTswana
wdTurkish
wdTurkmen
wdUkrainian
wdUrdu
wdUzbekCyrillic
wdUzbekLatin

wdVenda
wdVietnamese
wdWelsh
wdXhosa
wdZulu

Example

▶ [As it applies to the **Range** object.](#)

This example returns a list of synonyms for the selection's first meaning.

```
Slist = Selection.Range.SynonymInfo.SynonymList(Meaning:=1)
For i = 1 To UBound(Slist)
    MsgBox Slist(i)
Next i
```

▶ [As it applies to the **Application** and **Global** objects.](#)

This example returns a list of antonyms for the word "big" in U.S. English.

```
Alist = SynonymInfo(Word:="big", _
    LanguageID:=wdEnglishUS).AntonymList
For i = 1 To UBound(Alist)
    MsgBox Alist(i)
Next i
```



SynonymList Property

Returns a list of synonyms for a specified meaning of a word or phrase. The list is returned as an array of strings. Read-only **Variant**.

expression.**SynonymList**(*Meaning*)

expression Required. An expression that returns a [SynonymInfo](#) object.

Meaning Required **Variant**. The meaning as a string or an index number in the array of possible meanings.

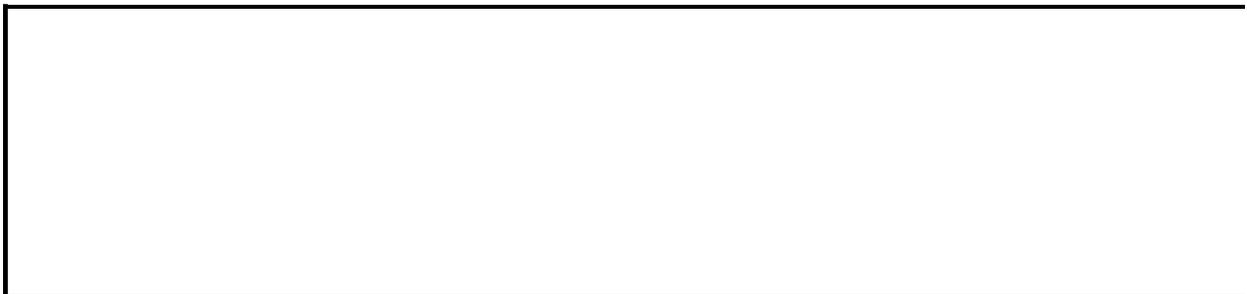
Example

This example returns a list of synonyms for the word "big," using the meaning "generous" in U.S. English.

```
Slist = SynonymInfo(Word:="big", LanguageID:=wdEnglishUS) _  
    .SynonymList(Meaning:="generous")  
For i = 1 To UBound(Slist)  
    MsgBox Slist(i)  
Next i
```

This example returns a list of synonyms for the second meaning of the selected word or phrase and displays these synonyms in the Immediate window of the Visual Basic editor. If there's no second meaning or if there are no synonyms, this is stated in a message box.

```
Set mySi = Selection.Range.SynonymInfo  
If mySi.MeaningCount >= 2 Then  
    synList = mySi.SynonymList(Meaning:=2)  
    For i = 1 To UBound(synList)  
        Debug.Print synList(i)  
    Next i  
Else  
    MsgBox "There is no second meaning for this word or phrase."  
End If
```



System Property

Returns a [System](#) object, which can be used to return system-related information and perform system-related tasks.

expression.**System**

expression Required. An expression that returns an [Application](#) object.

Example

This example returns information about the system.

```
processor = System.ProcessorType  
enviro = System.OperatingSystem
```

This example establishes a connection to a network drive.

```
System.Connect Path:=="\\Project\Info"
```



TabIndentKey Property

True if the TAB and BACKSPACE keys can be used to increase and decrease, respectively, the left indent of paragraphs and if the BACKSPACE key can be used to change right-aligned paragraphs to centered paragraphs and centered paragraphs to left-aligned paragraphs. Read/write **Boolean**.

expression.**TabIndentKey**

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

Example

This example sets Word so that the TAB and BACKSPACE keys set the left indent instead of inserting and deleting tabs.

```
Options.TabIndentKey = True
```

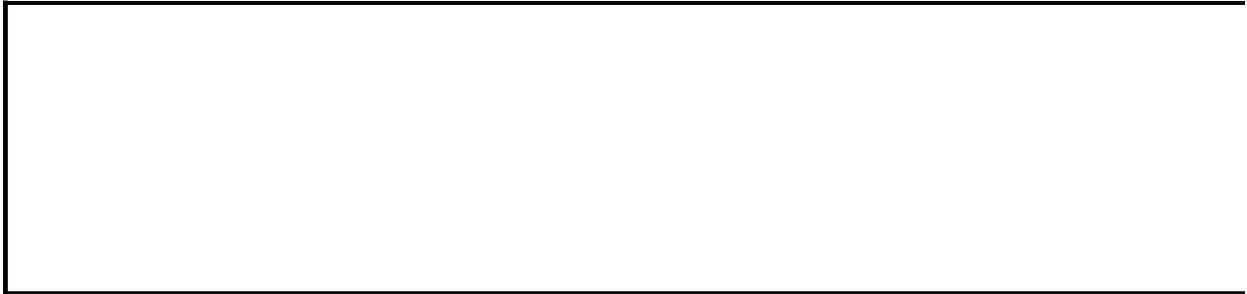


Table Property

Returns a **TableStyle** object representing properties that can be applied to a table using a table style.

expression.**Table**

expression Required. An expression that returns a [Style](#) object.

Example

This example creates a new table style that specifies a surrounding border and special borders and shading for only the first and last rows and the last column.

```
Sub NewTableStyle()  
    Dim styTable As Style  
  
    Set styTable = ActiveDocument.Styles.Add( _  
        Name:="TableStyle 1", Type:=wdStyleTypeTable)  
  
    With styTable.Table  
  
        'Apply borders around table, a double border to the heading  
        'a double border to the last column, and shading to last row  
        .Borders(wdBorderTop).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderBottom).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderLeft).LineStyle = wdLineStyleSingle  
        .Borders(wdBorderRight).LineStyle = wdLineStyleSingle  
  
        .Condition(wdFirstRow).Borders(wdBorderBottom) _  
            .LineStyle = wdLineStyleDouble  
  
        .Condition(wdLastColumn).Borders(wdBorderLeft) _  
            .LineStyle = wdLineStyleDouble  
  
        .Condition(wdLastRow).Shading _  
            .BackgroundColor = wdColorGray125  
  
    End With  
  
End Sub
```



↳ [Show All](#)

TabLeader Property

Returns or sets the character between entries and their page numbers in an index, table of authorities, table of contents, or table of figures. Read/write [WdTabLeader](#).

WdTabLeader can be one of these WdTabLeader constants.

wdTabLeaderDashes

wdTabLeaderDots

wdTabLeaderHeavy

wdTabLeaderLines

wdTabLeaderMiddleDot

wdTabLeaderSpaces

expression.**TabLeader**

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

Example

This example formats the tables of contents in Sales.doc to use a dotted tab leader.

```
For Each aTOC In Documents("Sales.doc").TablesOfContents
    aTOC.TabLeader = wdTabLeaderDots
Next aTOC
```

This example adds an index at the end of the active document. The page numbers are right-aligned with a dashed-line tab leader.

```
Set myRange = ActiveDocument.Range( _
    Start:=ActiveDocument.Content.End -1, _
    End:=ActiveDocument.Content.End -1)
ActiveDocument.Indexes.Add(Range:=myRange, Type:=wdIndexIndent, _
    RightAlignPageNumbers:=True).TabLeader = wdTabLeaderDashes
```



↳ [Show All](#)

TableDirection Property

Returns or sets the direction in which Microsoft Word orders cells in the specified table or row. Read/write [WdTableDirection](#).

WdTableDirection can be one of these WdTableDirection constants.

wdTableDirectionLtr

wdTableDirectionRtl

expression.**TableDirection**

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

Remarks

If the **TableDirection** property is set to **wdTableDirectionLtr**, the selected rows are arranged with the first column in the leftmost position. If the **TableDirection** property is set to **wdTableDirectionRtl**, the selected rows are arranged with the first column in the rightmost position.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Microsoft Word to order cells in the selected row from right to left.

```
Selection.Rows.TableDirection = _  
    wdTableDirectionRtl
```



TableGridlines Property

True if table gridlines are displayed. Read/write **Boolean**.

expression.**TableGridlines**

expression Required. An expression that returns a [View](#) object.

Example

This example displays table gridlines in the active window.

```
ActiveDocument.ActiveWindow.View.TableGridlines = True
```

This example shows table gridlines for the panes associated with window one in the [Windows](#) collection.

```
For Each myPane In Windows(1).Panes  
    myPane.View.TableGridlines = True  
Next myPane
```



TableName Property

Returns a **String** with the SQL query used to retrieve the records from the data source file attached to a mail merge document. 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 one of the objects in the Applies To list.

Example

This example checks to see if the Customers table is the name of the table in the attached data source. If not, it attaches the Customers table in the Northwind database.

```
Sub DataSourceTable()  
    With ActiveDocument.MailMerge  
        If InStr(1, .DataSource.TableName, "Customers") < 1 Then  
            .OpenDataSource Name:="C:\ProgramFiles\Microsoft Office\  
                "Samples\Northwind.mdb", LinkToSource:=True, _  
                AddToRecentFiles:=False, Connection:="TABLE Customer  
        End If  
    End With  
End Sub
```

Note This example uses the Visual Basic **InStr** function, which returns the position of the first character in the second string if it exists in the first string. A value of zero (0) is returned if the first string does not contain the second string. Setting the conditional value to less than one (1) indicates that the attached table is not named Customers.



Tables Property

Returns a [Tables](#) collection that represents all the tables in the specified cell, document, range, selection, or table. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a 5x5 table in the active document and then applies a predefined format to it.

```
Selection.Collapse Direction:=wdCollapseStart
Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _
NumRows:=5, NumColumns:=5)
myTable.AutoFormat Format:=wdTableFormatClassic2
```

This example inserts numbers and text into the first column of the first table in the active document.

```
num = 90
For Each acell In ActiveDocument.Tables(1).Columns(1).Cells
    acell.Range.Text = num & " Sales"
    num = num + 1
Next acell
```


TablesOfAuthorities Property

Returns a [TablesOfAuthorities](#) collection that represents the tables of authorities in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a table of authorities at the beginning of Sales.doc. The table of authorities compiles references from all categories.

```
Set myRange = Documents("Sales.doc").Range(Start:=0, End:=0)
Documents("Sales.doc").TablesOfAuthorities.Add Range:=myRange, _
    Category:=0, Passim:=True, IncludeCategoryHeader:=True
```

This example updates each table of authorities in the active document.

```
For Each myTOA In ActiveDocument.TablesOfAuthorities
    myTOA.Update
Next myTOA
```



TablesOfAuthoritiesCategories Property

Returns a [TablesOfAuthoritiesCategories](#) collection that represents the available table of authorities categories for the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example changes the name of the eighth item in the table of authorities category list for the active document.

```
ActiveDocument.TablesOfAuthoritiesCategories(8).Name = "Other case"
```

This example displays the name of the last table of authorities category if the category name has been changed.

```
last = ActiveDocument.TablesOfAuthoritiesCategories.Count  
If ActiveDocument.TablesOfAuthoritiesCategories(last) _  
    .Name <> "16" Then  
    MsgBox ActiveDocument.TablesOfAuthoritiesCategories(last).Name  
End If
```



TablesOfContents Property

Returns a [TablesOfContents](#) collection that represents the tables of contents in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a table of contents at the beginning of Sales.doc. The table of contents collects entry text from TC fields.

```
Set myRange = Documents("Sales.doc").Range(Start:=0, End:=0)
Documents("Sales.doc").TablesOfContents.Add Range:=myRange, _
    UseFields:=True, UseHeadingStyles:=False
```

This example updates the page numbers for items in the table of contents in the active document.

```
For Each myTOC In ActiveDocument.TablesOfContents
    myTOC.UpdatePageNumbers
Next myTOC
```



TablesOfFigures Property

Returns a [TablesOfFigures](#) collection that represents the tables of figures in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example adds a table of figures at the insertion point in the active document.

```
Selection.Collapse Direction:=wdCollapseStart
ActiveDocument.TablesOfFigures.Add Range:=Selection.Range, _
    Caption:=wdCaptionFigure
```

This example updates the contents of the first table of figures in Report.doc.

```
Documents("Report.doc").TablesOfFigures(1).Update
```



TabPosition Property

Returns or sets the tab position for the specified [ListLevel](#) object. Read/write **Single**.

expression.**TabPosition**

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

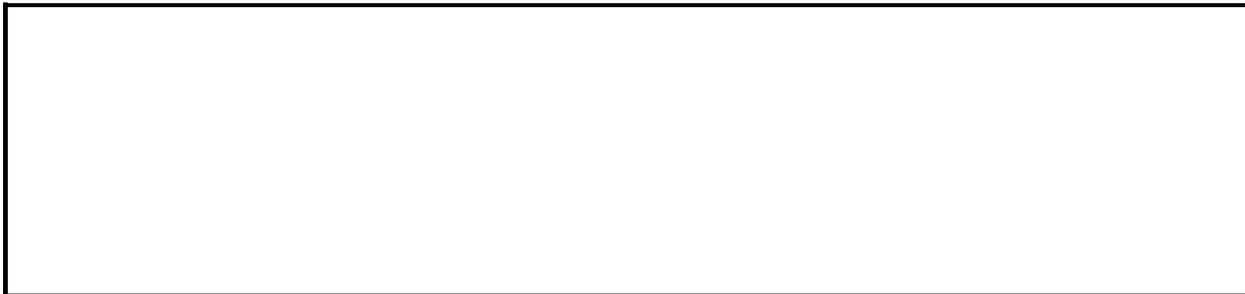
Example

This example sets each list level number so that it's indented 0.5 inch (36 points) from the previous level, and the tab is set at 0.25 inch (18 points) from the number.

```
r = 0
For Each lev In ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(1).ListLevels
    lev.Alignment = wdListLevelAlignLeft
    lev.NumberPosition = r
    lev.TrailingCharacter = wdTrailingTab
    lev.TabPosition = r + 18
    r = r + 36
Next lev
```

This example sets the variable `myltemp` to the first numbered list template, and then it sets the tab position at one inch. The list template is then applied to the selection.

```
Set myltemp = ListGalleries(wdNumberGallery).ListTemplates(1)
myltemp.ListLevels(1).TabPosition = InchesToPoints(1)
Selection.Range.ListFormat.ApplyListTemplate ListTemplate:=myltemp
```



TabStops Property

Returns or sets a [TabStops](#) collection that represents all the custom tab stops for the specified paragraphs. Read/write.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

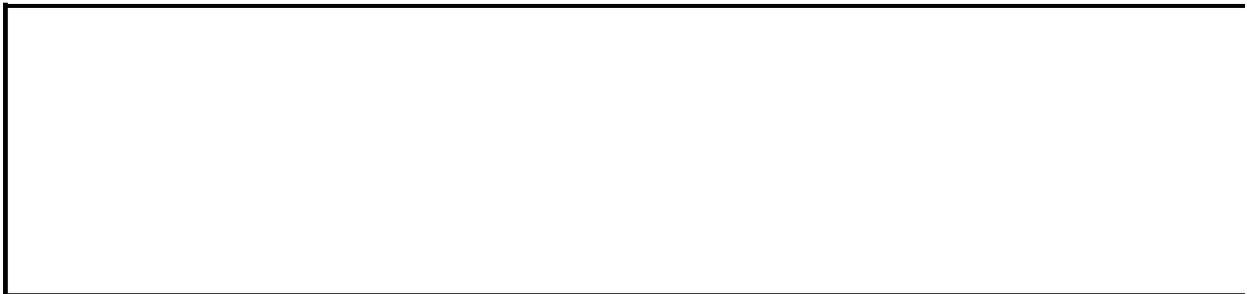
Example

This example adds a centered tab stop at 2 inches to all the paragraphs in the active document. The **InchesToPoints** method is used to convert inches to points.

```
With ActiveDocument.Paragraphs.TabStops  
    .Add Position:= InchesToPoints(2), Alignment:= wdAlignTabCenter  
End With
```

This example sets the tab stops for every paragraph in the document to match the tab stops in the first paragraph.

```
Set para1Tabs = ActiveDocument.Paragraphs(1).TabStops  
ActiveDocument.Paragraphs.TabStops = para1Tabs
```



↳ [Show All](#)

Target Property

▶ [Target property as it applies to the **Browser** object.](#)

Returns or sets the document item that the [Previous](#) and [Next](#) methods locate.
Read/write [WdBrowseTarget](#).

WdBrowseTarget can be one of these WdBrowseTarget constants.

wdBrowseComment

wdBrowseEdit

wdBrowseEndnote

wdBrowseField

wdBrowseFind

wdBrowseFootnote

wdBrowseGoTo

wdBrowseGraphic

wdBrowseHeading

wdBrowsePage

wdBrowseSection

wdBrowseTable

expression.**Target**

expression Required. An expression that returns a [Browser](#) object.

▶ [Target property as it applies to the **Hyperlink** object.](#)

Returns or sets the name of the frame or window in which to load the hyperlink.
Read/write **String**.

expression.**Target**

expression Required. An expression that returns a [Hyperlink](#) object.

Example

▶ [As it applies to the **Browser** object.](#)

This example moves the insertion point to the next comment in the active document.

```
With Application.Browser
    .Target = wdBrowseComment
    .Next
End With
```

▶ [As it applies to the **Hyperlink** object.](#)

This example sets the specified hyperlink to open in a new browser window.

```
ActiveDocument.Hyperlinks(1).Target = "_blank"
```

This example sets the specified hyperlink to open in the frame called "left."

```
ActiveDocument.Hyperlinks(1).Target = "left"
```



↳ [Show All](#)

TargetBrowser Property

Sets or returns an [MsoTargetBrowser](#) constant representing the target browser for documents viewed in a Web browser. Read/write.

MsoTargetBrowser can be one of these **MsoTargetBrowser** constants.

msoTargetBrowserIE4 Microsoft Internet Explorer 4.0.

msoTargetBrowserIE5 Internet Explorer 5.

msoTargetBrowserIE6 Internet Explorer 6.

msoTargetBrowserV3 Netscape Navigator 3.

msoTargetBrowserV4 Netscape Navigator 4.

expression.**TargetBrowser**

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

Remark

The **TargetBrowser** property sets the [BrowserLevel](#) property, but **BrowserLevel** is only important if the [DisableFeatures](#) property is set to **True**. Otherwise, it is ignored. The **TargetBrowser** property, however, is not ignored and sets the browser level for all Web documents or for a single Web document.

Example

This example sets the target browser for the active document to Microsoft Internet Explorer 6 if the current target browser is an earlier version.

```
Sub SetWebBrowser()  
    With ActiveDocument.WebOptions  
        If .TargetBrowser < msoTargetBrowserIE6 Then  
            .TargetBrowser = msoTargetBrowserIE6  
        End If  
    End With  
End Sub
```

This example sets the target browser for all documents to Internet Explorer 6.

```
Sub GlobalTargetBrowser()  
    Application.DefaultWebOptions _  
        .TargetBrowser = msoTargetBrowserIE6  
End Sub
```



TaskPanes Property

Returns a **TaskPanes** object that represents the most commonly performed tasks in Microsoft Word.

expression.**TaskPanes**

expression Required. An expression that returns an **Application** object.

Example

The following example displays the task pane that contains information about formatting in a document.

```
Sub showFormatting()  
    Application.TaskPanes.Item(wdTaskPaneFormatting).Visible = True  
End Sub
```



Tasks Property

Returns a [Tasks](#) collection that represents all the applications that are running.

expression.**Tasks**

expression Required. An expression that returns an [Application](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the calculator. If the calculator is not already running, then Word starts the task and then displays the calculator.

```
If Tasks.Exists("Calculator") Then
    With Tasks("Calculator")
        .Activate
        .WindowState = wdWindowStateNormal
    End With
Else
    Shell "calc.exe"
    Tasks("Calculator").WindowState = wdWindowStateNormal
End If
```

This example checks to see whether Microsoft Excel is currently running. If the task is running, the example activates Microsoft Excel; otherwise, a message box is displayed.

```
If Tasks.Exists("Microsoft Excel") = True Then
    With Tasks("Microsoft Excel")
        .Activate
        .WindowState = wdWindowStateMaximize
    End With
Else
    MsgBox "Microsoft Excel is not currently running."
End If
```



Templates Property

Returns a [Templates](#) collection that represents all the available templates — global templates as well as those attached to open documents.

expression.**Templates**

expression Required. An expression that returns an [Application](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the name of each template in the **Templates** collection.

```
Count = 1
For Each aTemplate In Templates
    MsgBox aTemplate.Name & " is template number " & Count
    Count = Count + 1
Next aTemplate
```

In this example, if template one is a global template, its path is stored in thePath. The **ChDir** statement is used to make the folder with the path stored in thePath the current folder. When this change is made, the **Open** dialog box is displayed.

```
If Templates(1).Type = wdGlobalTemplate Then
    thePath = Templates(1).Path
    If thePath <> "" Then ChDir thePath
    Dialogs(wdDialogFileOpen).Show
End If
```



Text Property

-
Range or **Selection** object: Returns or sets the text in the specified range or selection. Read/write **String**.

Find or **Replacement** object: Returns or sets the text to find or replace in the specified range or selection. Read/write **String**.

Remarks

The **Text** property returns the plain, unformatted text of the selection or range. When you set this property, the text of the range or selection is replaced.

Example

This example displays the text in the selection. If nothing is selected, the character following the insertion point is displayed.

```
MsgBox Selection.Text
```

This example replaces the first word in the active document with "Dear."

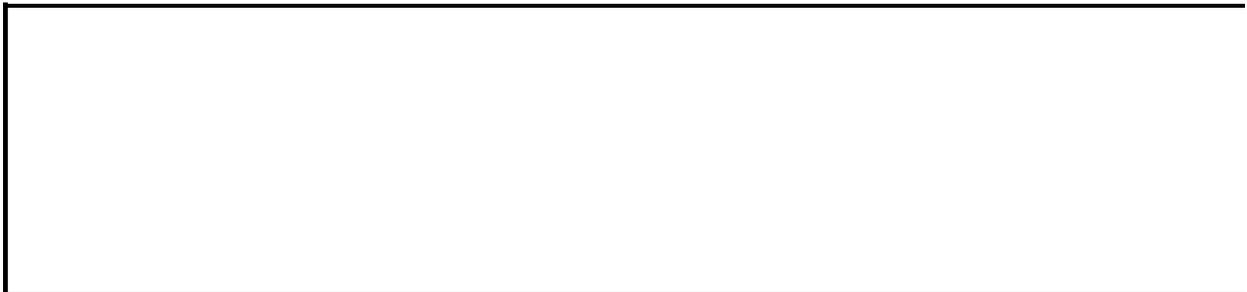
```
Set myRange = ActiveDocument.Words(1)
myRange.Text = "Dear "
```

This example inserts 10 lines of text into a new document.

```
Documents.Add
For i = 1 To 10
    Selection.Text = "Line" & Str(i) & Chr(13)
    Selection.MoveDown Unit:=wdParagraph, Count:=1
Next i
```

This example replaces "Hello" with "Goodbye" in the active document.

```
Set myRange = ActiveDocument.Content
With myRange.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "Hello"
    .Replacement.Text = "Goodbye"
    .Execute Replace:=wdReplaceAll
End With
```



TextColumns Property

Returns a [TextColumns](#) collection that represents the set of text columns for the specified [PageSetup](#) object.

expression.**TextColumns**

expression Required. An expression that returns a [PageSetup](#) object.

Remarks

There will always be at least one text column in the collection. When you create new text columns, you're adding to a collection of one column.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates four evenly-spaced text columns that are applied to section two in the active document.

```
With ActiveDocument.Sections(2).PageSetup.TextColumns  
    .SetCount NumColumns:=3  
    .Add EvenlySpaced:=True  
End With
```

This example creates a document with two text columns. The first column is 1.5 inches wide and the second is 3 inches wide.

```
Set myDoc = Documents.Add  
With myDoc.PageSetup.TextColumns  
    .SetCount NumColumns:=1  
    .Add Width:=InchesToPoints(3)  
End With  
With myDoc.PageSetup.TextColumns(1)  
    .Width = InchesToPoints(1.5)  
    .SpaceAfter = InchesToPoints(0.5)  
End With
```



TextEffect Property

-

Returns a [TextEffectFormat](#) object that contains text-effect formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent WordArt and to **InlineShape** objects. Read-only.

Example

This example sets the font style to bold for shape three on myDocument if the shape is WordArt.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontBold = True
    End If
End With
```



↳ [Show All](#)

TextEncoding Property

Returns or sets the code page, or character set, that Microsoft Word uses for a document saved as an encoded text file. Read/write [MsoEncoding](#).

MsoEncoding can be one of these MsoEncoding constants.

msoEncodingArabic

msoEncodingArabicASMO

msoEncodingArabicAutoDetect Not used with this property.

msoEncodingArabicTransparentASMO

msoEncodingAutoDetect Not used with this property.

msoEncodingBaltic

msoEncodingCentralEuropean

msoEncodingCyrillic

msoEncodingCyrillicAutoDetect Not used with this property.

msoEncodingEBCDICArabic

msoEncodingEBCDICDenmarkNorway

msoEncodingEBCDICFinlandSweden

msoEncodingEBCDICFrance

msoEncodingEBCDICGermany

msoEncodingEBCDICGreek

msoEncodingEBCDICGreekModern

msoEncodingEBCDICHebrew

msoEncodingEBCDICIcelandic

msoEncodingEBCDICInternational

msoEncodingEBCDICItaly

msoEncodingEBCDICJapaneseKatakanaExtended

msoEncodingEBCDICJapaneseKatakanaExtendedAndJapanese

msoEncodingEBCDICJapaneseLatinExtendedAndJapanese

msoEncodingEBCDICKoreanExtended

msoEncodingEBCDICKoreanExtendedAndKorean
msoEncodingEBCDICLatinAmericaSpain
msoEncodingEBCDICMultilingualROECELatin2
msoEncodingEBCDICRussian
msoEncodingEBCDICSerbianBulgarian
msoEncodingEBCDICSimplifiedChineseExtendedAndSimplifiedChinese
msoEncodingEBCDICThai
msoEncodingEBCDICTurkish
msoEncodingEBCDICTurkishLatin5
msoEncodingEBCDICUnitedKingdom
msoEncodingEBCDICUSCanada
msoEncodingEBCDICUSCanadaAndJapanese
msoEncodingEBCDICUSCanadaAndTraditionalChinese
msoEncodingEUCChineseSimplifiedChinese
msoEncodingEUCJapanese
msoEncodingEUCKorean
msoEncodingEUCTaiwaneseTraditionalChinese
msoEncodingEuropa3
msoEncodingExtAlphaLowercase
msoEncodingGreek
msoEncodingGreekAutoDetect Not used with this property.
msoEncodingHebrew
msoEncodingHZGBSimplifiedChinese
msoEncodingIA5German
msoEncodingIA5IRV
msoEncodingIA5Norwegian
msoEncodingIA5Swedish
msoEncodingISO2022CNSimplifiedChinese
msoEncodingISO2022CNTraditionalChinese
msoEncodingISO2022JPJISX02011989
msoEncodingISO2022JPJISX02021984
msoEncodingISO2022JPNoHalfwidthKatakana
msoEncodingISO2022KR

msoEncodingISO6937NonSpacingAccent
msoEncodingISO885915Latin9
msoEncodingISO88591Latin1
msoEncodingISO88592CentralEurope
msoEncodingISO88593Latin3
msoEncodingISO88594Baltic
msoEncodingISO88595Cyrillic
msoEncodingISO88596Arabic
msoEncodingISO88597Greek
msoEncodingISO88598Hebrew
msoEncodingISO88599Turkish
msoEncodingJapaneseAutoDetect Not used with this property.
msoEncodingJapaneseShiftJIS
msoEncodingKOI8R
msoEncodingKOI8U
msoEncodingKorean
msoEncodingKoreanAutoDetect Not used with this property.
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 Not used with this property.
msoEncodingSimplifiedChineseGBK
msoEncodingT61
msoEncodingTaiwanCNS
msoEncodingTaiwanEten
msoEncodingTaiwanIBM5550
msoEncodingTaiwanTCA
msoEncodingTaiwanTeleText
msoEncodingTaiwanWang
msoEncodingThai
msoEncodingTraditionalChineseAutoDetect Not used with this property.
msoEncodingTraditionalChineseBig5
msoEncodingTurkish
msoEncodingUnicodeBigEndian
msoEncodingUnicodeLittleEndian
msoEncodingUSASCII
msoEncodingUTF7
msoEncodingUTF8

msoEncodingVietnamese
msoEncodingWestern

expression.**TextEncoding**

expression Required. An expression that returns a [Document](#) object.

Remarks

The **TextEncoding** property sets text encoding separately from HTML encoding, which you can set using the [Encoding](#) property. To set text encoding for all documents saved as text files, use the [DefaultTextEncoding](#) property.

Example

This example sets the text encoding for the active document to Japanese if it is saved as a text file.

```
Sub EncodeText()  
    ActiveDocument.TextEncoding = msoEncodingJapaneseShiftJIS  
End Sub
```



TextFrame Property

-

Returns a [TextFrame](#) object that contains the text for the specified shape.

Example

This example adds a rectangle to myDocument, adds text to the rectangle, and sets the margins for the text frame.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeRectangle, _
    0, 0, 250, 140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 0
    .MarginLeft = 100
    .MarginRight = 0
    .MarginTop = 20
End With
```



TextInput Property

Returns a [TextInput](#) object that represents a text form field.

expression.**TextInput**

expression Required. An expression that returns a [FormField](#) object.

Remarks

If the **TextInput** property is applied to a **FormField** object that isn't a drop-down form field, the property won't fail, but the [Valid](#) property for the returned object will be **False**.

Use the [Result](#) property with the **FormField** object to return or set the contents of a [TextInput](#) object, as follows:

```
ActiveDocument.FormFields("Text1").Result = "John Doe"
```

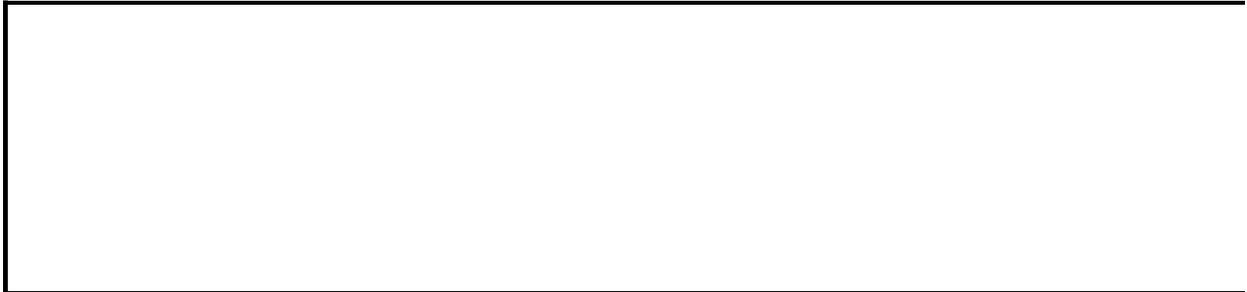
Example

This example protects the active document for forms and deletes the contents of the form field named "Text1."

```
ActiveDocument.Protect Type:=wdAllowOnlyFormFields  
ActiveDocument.FormFields("Text1").TextInput.Clear
```

If the first form field in the active document is a text form field that accepts regular text, this example sets the contents of the form field.

```
Set myField = ActiveDocument.FormFields(1)  
If myField.Type = wdFieldFormTextInput And _  
    myField.TextInput.Type = wdRegularText Then  
    myField.Result = "Hello"  
End If
```



↳ [Show All](#)

TextLineEnding Property

Returns or sets a [WdLineEndingType](#) constant indicating how Microsoft Word marks the line and paragraph breaks in documents saved as text files.
Read/write.

WdLineEndingType can be one of these WdLineEndingType constants.

wdCRLF

wdCROnly

wdLF CR Default

wdLFOnly

wdLSPS

expression.**TextLineEnding**

expression Required. An expression that returns a [Document](#) object.

Example

This example sets the active document to enter a carriage return for line and paragraph breaks when it is saved as a text file.

```
Sub LineEndings()  
    ActiveDocument.TextLineEnding = wdCROnly  
End Sub
```



TextPosition Property

Returns or sets the position (in points) for the second line of wrapping text for the specified [ListLevel](#) object. Read/write **Single**.

expression.**TextPosition**

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

Example

This example sets the indentation for all levels of the first outline-numbered list template. Each list level number is indented 0.5 inch (36 points) from the previous level, the tab is set at 0.25 inch (18 points) from the number, and wrapping text is indented 0.25 inch (18 points) from the number.

```
r = 0
For Each lev In ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(1).ListLevels
    lev.Alignment = wdListLevelAlignLeft
    lev.NumberPosition = r
    lev.TrailingCharacter = wdTrailingTab
    lev.TabPosition = r + 18
    lev.TextPosition = r + 18
    r = r + 36
Next lev
```



TextRange Property

Returns a [Range](#) object that represents the text in the specified text frame.

expression.**TextRange**

expression Required. An expression that returns a [TextFrame](#) object.

Example

This example adds a text box to the active document and then adds text to the text box.

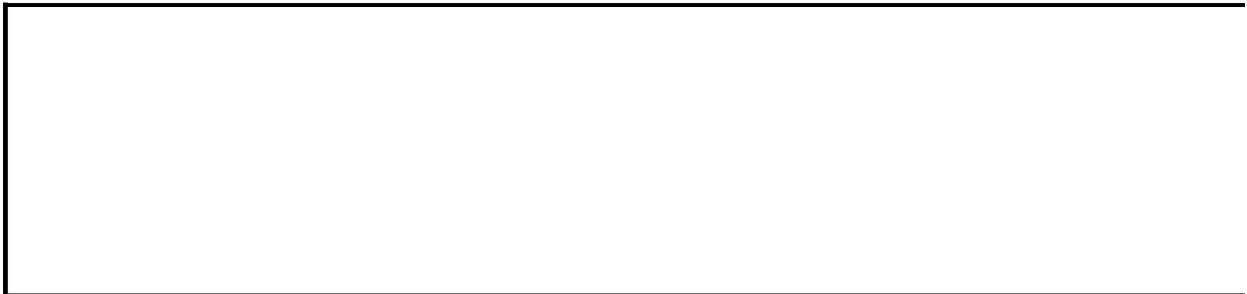
```
Set myTBox = ActiveDocument.Shapes _  
    .AddTextBox(Orientation:=msoTextOrientationHorizontal, _  
        Left:=100, Top:=100, Width:=300, Height:=200)  
myTBox.TextFrame.TextRange = "Test Box"
```

This example adds text to TextBox 1 in the active document.

```
ActiveDocument.Shapes("TextBox 1").TextFrame.TextRange _  
    .InsertAfter("New Text")
```

This example returns the text from TextBox 1 in the active document and displays it in a message box.

```
MsgBox ActiveDocument.Shapes("TextBox 1").TextFrame.TextRange.Text
```



TextRetrievalMode Property

-
Returns a [TextRetrievalMode](#) object that controls how text is retrieved from the specified **Range**. Read/write.

Example

This example retrieves the selected text (excluding any hidden text) and inserts it at the beginning of the third paragraph in the active document.

```
If Selection.Type = wdSelectionNormal Then
    Set Range1 = Selection.Range
    Range1.TextRetrievalMode.IncludeHiddenText = False
    Set Range2 = ActiveDocument.Paragraphs(2).Range
    Range2.InsertAfter Range1.Text
End If
```

This example retrieves and displays the first three paragraphs as they appear in outline view.

```
Set myRange = ActiveDocument.Range(Start:=ActiveDocument _
    .Paragraphs(1).Range.Start, _
    End:=ActiveDocument.Paragraphs(3).Range.End)
myRange.TextRetrievalMode.ViewType = wdOutlineView
MsgBox myRange.Text
```

This example excludes field codes and hidden text from the range that refers to the selected text. The example then displays the text in a message box.

```
If Selection.Type = wdSelectionNormal Then
    Set aRange = Selection.Range
    With aRange.TextRetrievalMode
        .IncludeHiddenText = False
        .IncludeFieldCodes = False
    End With
    MsgBox aRange.Text
End If
```



TextShape Property

Returns a [Shape](#) object that represents the shape of the text box associated with a diagram node.

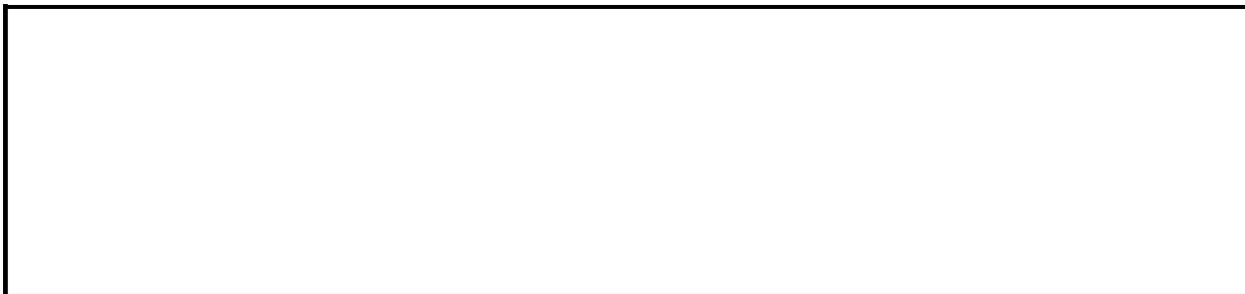
expression.**TextShape**

expression Required. An expression that returns a [DiagramNode](#) object.

Example

This example adds child nodes to a parent node and displays text in the parent node indicating the number of child nodes created.

```
Sub CountChildNodes()  
    Dim shpDiagram As Shape  
    Dim dgnNode As DiagramNode  
    Dim shpText As Shape  
    Dim intCount As Integer  
  
    'Add radial diagram to the current document  
    Set shpDiagram = ThisDocument.Shapes.AddDiagram _  
        (Type:=msoDiagramRadial, Left:=10, _  
         Top:=15, Width:=400, Height:=475)  
  
    'Add first node to the diagram  
    Set dgnNode = shpDiagram.DiagramNode.Children.AddNode  
  
    'Add three child nodes  
    For intCount = 1 To 3  
        dgnNode.Children.AddNode  
    Next intCount  
  
    'Add a text box for each node in the diagram  
    For intCount = 1 To 4  
        Set shpText = shpDiagram.DiagramNode.Children(1).TextShape  
        shpText.TextFrame.TextRange.Text = Str(intCount)  
    Next intCount  
End Sub
```



TextToDisplay Property

Returns or sets the specified hyperlink's visible text in a document. Read/write **String**.

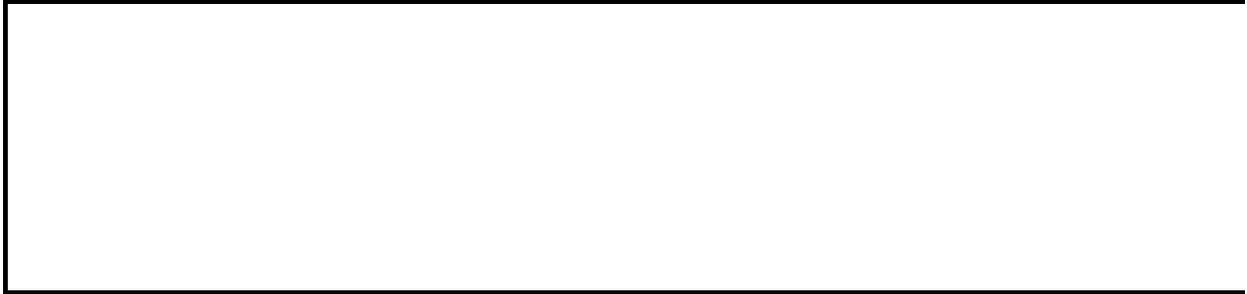
expression.**TextToDisplay**

expression Required. An expression that returns a [Hyperlink](#) object.

Example

This example sets the display text for the first hyperlink in the active document.

```
ActiveDocument.Hyperlinks(1).TextToDisplay = _  
    "Follow this link for more information..."
```



↳ [Show All](#)

Texture Property

Returns or sets the shading texture for the specified object. Read/write [WdTextureIndex](#).

WdTextureIndex can be one of these WdTextureIndex constants.

wdTexture10Percent

wdTexture12Pt5Percent

wdTexture15Percent

wdTexture17Pt5Percent

wdTexture20Percent

wdTexture22Pt5Percent

wdTexture25Percent

wdTexture27Pt5Percent

wdTexture2Pt5Percent

wdTexture30Percent

wdTexture32Pt5Percent

wdTexture35Percent

wdTexture37Pt5Percent

wdTexture40Percent

wdTexture42Pt5Percent

wdTexture45Percent

wdTexture47Pt5Percent

wdTexture50Percent

wdTexture52Pt5Percent

wdTexture55Percent

wdTexture57Pt5Percent

wdTexture5Percent

wdTexture60Percent

wdTexture62Pt5Percent

wdTexture65Percent
wdTexture67Pt5Percent
wdTexture70Percent
wdTexture72Pt5Percent
wdTexture75Percent
wdTexture77Pt5Percent
wdTexture7Pt5Percent
wdTexture80Percent
wdTexture82Pt5Percent
wdTexture85Percent
wdTexture87Pt5Percent
wdTexture90Percent
wdTexture92Pt5Percent
wdTexture95Percent
wdTexture97Pt5Percent
wdTextureCross
wdTextureDarkCross
wdTextureDarkDiagonalCross
wdTextureDarkDiagonalDown
wdTextureDarkDiagonalUp
wdTextureDarkHorizontal
wdTextureDarkVertical
wdTextureDiagonalCross
wdTextureDiagonalDown
wdTextureDiagonalUp
wdTextureHorizontal
wdTextureNone
wdTextureSolid
wdTextureVertical

expression.**Texture**

expression Required. An expression that returns a [Shading](#) object.

Example

This example sets a range that references the first paragraph in the active document and then applies a grid texture to that range.

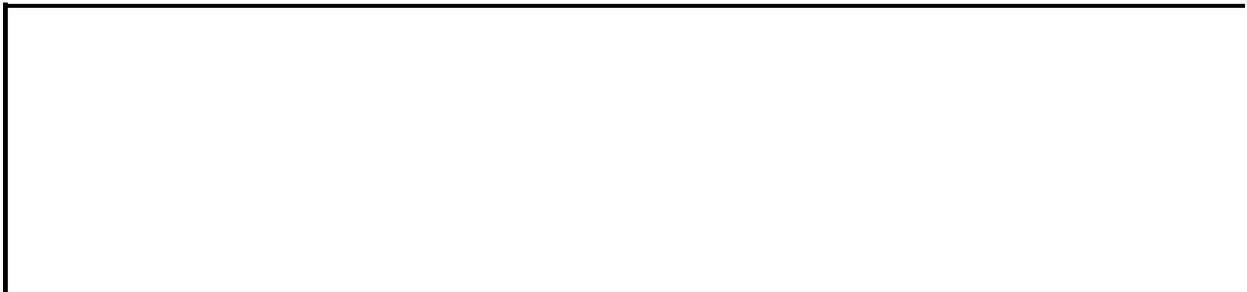
```
Set myRange = ActiveDocument.Paragraphs(1).Range  
myRange.Shading.Texture = wdTextureCross
```

This example adds a table at the insertion point and then applies a vertical line texture to the first row in the table.

```
Selection.Collapse Direction:=wdCollapseStart  
Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _  
    NumRows:=2, NumColumns:=2)  
myTable.Rows(1).Shading.Texture = wdTextureVertical
```

This example applies 10 percent shading to the first word in the active document.

```
ActiveDocument.Words(1).Shading.Texture = wdTexture10Percent
```



TextureName Property

Returns the name of the custom texture file for the specified fill. Read-only **String**.

expression.**TextureName**

expression Required. An expression that returns a [FillFormat](#) object.

Remarks

Use the [UserTextured](#) method to set the texture file for the fill.

Example

This example adds an oval to the active document. If the second shape in the document has a user-defined textured fill, the new oval will have the same fill as shape two. If shape two has any other type of fill, the new oval will have a green marble fill. This example assumes that the active document already has at least two shapes.

```
With ActiveDocument.Shapes
    Set newFill = .AddShape(msoShapeOval, 0, 0, 200, 90).Fill
    With .Item(2).Fill
        If.TextureType = msoTextureUserDefined Then
            newFill.UserTextured .TextureName
        Else
            newFill.PresetTextured msoTextureGreenMarble
        End If
    End With
End With
```



↳ [Show All](#)

TextureType Property

Returns the texture type for the specified fill. Read-only [MsoTextureType](#).

MsoTextureType can be one of these MsoTextureType constants.

msoTexturePreset

msoTextureTypeMixed

msoTextureUserDefined

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](#), [UserPicture](#), or [UserTextured](#) method to set the texture type for the fill.

Example

This example changes the fill for all shapes in the active document with a custom textured fill to a canvas fill.

```
For Each s In ActiveDocument.Shapes
  With s.Fill
    If .TextureType = msoTextureUserDefined Then
      .PresetTextured msoTextureCanvas
    End If
  End With
Next
```



TextWrap Property

True if document text wraps around the specified frame. Read/write **Boolean**.

expression.**TextWrap**

expression Required. An expression that returns a [Frame](#) object.

Example

This example causes text to not wrap around the first frame in the active document.

```
If ActiveDocument.Frames.Count >= 1 Then
    ActiveDocument.Frames(1).TextWrap = False
End If
```

This example causes text to wrap around all frames in the active document.

```
For Each aFrame In ActiveDocument.Frames
    aFrame.TextWrap = True
Next aFrame
```



↳ [Show All](#)

ThemeName Property

-

Returns or sets the name of the [theme](#) plus any theme formatting options to use for new e-mail messages. Read/write **String**.

Remarks

For an explanation of the value returned by this property, see the *Name* argument of the [ApplyTheme](#) method. The value returned by this property may not correspond to the theme's display name as it appears in the **Theme** dialog box. To return a theme's display name, use the [ActiveThemeDisplayName](#) property.

You can also use the [GetDefaultTheme](#) and [SetDefaultTheme](#) methods to return and set the default theme for new e-mail messages.

Example

This example sets Microsoft Word to use the Blueprint theme with Vivid Colors for all new e-mail messages.

```
Application.EmailOptions.ThemeName = "blueprnt 100"
```



ThreeD Property

-

Returns a [ThreeDFormat](#) object that contains 3-D – effect formatting properties for the specified shape. Read-only.

Example

This example sets the depth, extrusion color, extrusion direction, and lighting direction for the 3-D effects applied to shape one on myDocument.

```
Set myDocument = ActiveDocument
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .Depth = 50
    ' RGB value for purple
    .ExtrusionColor.RGB = RGB(255, 100, 255)
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```



TintAndShade Property

Returns a **Single** that represents the lightening or darkening of a specified shape's color. Read/write.

expression.**TintAndShade**

expression Required. An expression that returns a [ColorFormat](#) object.

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.Shapes _  
        .AddShape(Type:=msoShapeHeart, Left:=150, _  
            Top:=150, Width:=250, Height:=250)  
    With shpHeart.Fill.ForeColor  
        .RGB = RGB(Red:=255, Green:=28, Blue:=0)  
        .TintAndShade = 0.3  
    End With  
End Sub
```



Title Property

Returns a **String** representing the title of a Web style sheet. Read/write.

expression.**Title**

expression Required. An expression that returns a [Sylesheet](#) object.

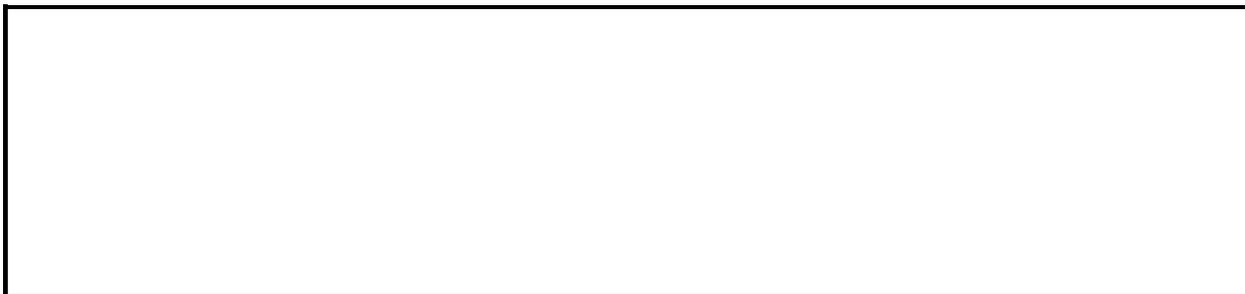
Example

This example assigns titles to the first three Web style sheets attached to the active document. This example assumes that there are three style sheets attached to the active document.

```
Sub AssignCSSTitle()  
    ActiveDocument.StyleSheets.Item(1).Title = "New Look Stylesheet"  
    ActiveDocument.StyleSheets.Item(2).Title = "Standard Web Stylesheet"  
    ActiveDocument.StyleSheets.Item(3).Title = "Definitions Stylesheet"  
End Sub
```

This example creates a list of Web style sheets attached to the active document and places the list in a new document. This example assumes there are one or more Web style sheets attached to the active document.

```
Sub CSSTitles()  
    Dim docNew As Document  
    Dim styCSS As StyleSheet  
  
    Set docNew = Documents.Add  
  
    With docNew.Range(Start:=0, End:=0)  
        .InsertAfter "CSS Name : Assigned to " & ThisDocument.Name _  
            & vbTab & "Title"  
        .InsertParagraphAfter  
        For Each styCSS In ThisDocument.StyleSheets  
            .InsertAfter styCSS.Name & vbTab & styCSS.Title  
            .InsertParagraphAfter  
        Next styCSS  
        .ConvertToTable  
    End With  
End Sub
```



↳ [Show All](#)

Top Property

▸ [Top property as it applies to the **Shape** and **ShapeRange** objects.](#)

Returns or sets the vertical position of the specified shape or shape range in points. Read/write **Single**.

expression.**Top**

expression Required. An expression that returns one of the above objects.

Remarks

The position of a shape is measured from the upper-left corner of the shape's bounding box to the shape's anchor. The [RelativeVerticalPosition](#) property controls whether the shape's anchor is positioned alongside the line, the paragraph, the margin, or the edge of the page.

For a [ShapeRange](#) object that contains more than one shape, the **Top** property sets the vertical position of each shape.

▶ [Top property as it applies to the Application, Task, and Window objects.](#)

Returns or sets the vertical position of the active document (for the [Application](#) object) or the specified task or window, in points. Read/write **Long**.

expression.**Top**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Application** object.](#)

This example positions the Word application window 100 points from the top of the screen.

```
Application.WindowState = wdWindowStateNormal
Application.Top = 100
```

▶ [As it applies to the **Shape** object.](#)

This example sets the vertical position of the first shape in the active document to 1 inch from the top of the page.

```
With ActiveDocument.Shapes(1)
    .RelativeVerticalPosition = wdRelativeVerticalPositionPage
    .Top = InchesToPoints(1)
End With
```

This example sets the vertical position of the first and second shapes in the active document to 1 inch from the top of the page.

```
With ActiveDocument.Shapes.Range(Array(1, 2))
    .RelativeVerticalPosition = wdRelativeVerticalPositionPage
    .Top = InchesToPoints(1)
End With
```

▶ [As it applies to the **Task** object.](#)

This example starts the Calculator and positions its window 100 points from the top of the screen.

```
Shell "Calc.exe"
With Tasks("Calculator")
    .WindowState = wdWindowStateNormal
    .Top = 100
End With
```



TopLevelTables Property

-

Returns a [Tables](#) collection that represents the tables at the outermost nesting level in the current range or selection. Read-only.

Remarks

This method returns a collection containing only those tables at the outermost nesting level within the context of the current range or selection. These tables may not be at the outermost nesting level within the entire set of nested tables.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example creates a new document, creates a nested table with three levels, and then fills the first cell of each table with its nesting level. The example selects the second column of the second-level table and then selects the first of the top-level tables in this selection. The innermost table is selected, even though it isn't a top-level table within the context of the entire set of nested tables.

```
Documents.Add
ActiveDocument.Tables.Add Selection.Range, _
    3, 3, wdWord9TableBehavior, wdAutoFitContent
With ActiveDocument.Tables(1).Range
    .Copy
    .Cells(1).Range.Text = .Cells(1).NestingLevel
    .Cells(5).Range.PasteAsNestedTable
    With .Cells(5).Tables(1).Range
        .Cells(1).Range.Text = .Cells(1).NestingLevel
        .Cells(5).Range.PasteAsNestedTable
        With .Cells(5).Tables(1).Range
            .Cells(1).Range.Text = _
                .Cells(1).NestingLevel
        End With
        .Columns(2).Select
        Selection.TopLevelTables(1).Select
    End With
End With
```



TopMargin Property

Returns or sets the distance (in points) between the top edge of the page and the top boundary of the body text. Read/write **Single**.

expression.**TopMargin**

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

Example

This example sets the top margin to 72 points (1 inch) for the first section in the active document.

```
ActiveDocument.Sections(1).PageSetup.TopMargin = 72
```

This example creates a new custom label and sets several properties, including the top margin, and then it creates a new document using the custom labels.

```
Set newlbl = Application.MailingLabel. _  
    CustomLabels.Add(Name:="My Label")  
With newlbl  
    .Height = InchesToPoints(1.25)  
    .NumberAcross = 2  
    .NumberDown = 7  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0)  
    .TopMargin = InchesToPoints(1)  
    .Width = InchesToPoints(4.25)  
End With  
Application.MailingLabel.CreateNewDocument Name:="My Label"
```



TopPadding Property

Returns or sets the amount of space (in points) to add above the contents of a single cell or all the cells in a table. Read/write **Single**.

expression.**TopPadding**

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

Remarks

The setting of the **TopPadding** property for a single cell overrides the setting of the **TopPadding** property for the entire table.

Example

This example sets the top padding for the first table in the active document to 40 pixels.

```
ActiveDocument.Tables(1).TopPadding = _  
    PixelsToPoints(40, True)
```



Tracking Property

Returns or sets the ratio of the horizontal space allotted to each character in the specified WordArt in relation to the width of the character. Can be a value from 0 (zero) through 5. (Large values for this property specify ample space between characters; values less than 1 can produce character overlap.) Read/write **Single**.

expression.**Tracking**

expression Required. An expression that returns a [TextEffectFormat](#) object.

Remarks

The following table gives the values of the **Tracking** property that correspond to the settings available in the user interface.

User interface setting	Equivalent Tracking property value
Very Tight	0.8
Tight	0.9
Normal	1.0
Loose	1.2
Very Loose	1.5

Example

This example adds WordArt that contains the text "Test" to the active document and specifies that the characters be very tightly spaced.

```
Set newWordArt = ActiveDocument.Shapes.AddTextEffect( _  
    PresetTextEffect:=msoTextEffect1, Text:="Test", _  
    FontName:="Arial Black", FontSize:=36, FontBold:=False, _  
    FontItalic:=False, Left:=100, Top:=100)  
newWordArt.TextEffect.Tracking = 0.8
```



TrackRevisions Property

-

True if changes are tracked in the specified document. Read/write **Boolean**.

Example

This example sets the active document so that it tracks changes and makes them visible on the screen.

```
With ActiveDocument
    .TrackRevisions = True
    .ShowRevisions = True
End With
```

This example inserts text if change tracking isn't enabled.

```
If ActiveDocument.TrackRevisions = False Then
    Selection.InsertBefore "new text"
End If
```



TrackStatus Property

-
True if a mail message is sent back to the original sender each time the routed document is forwarded. Read/write **Boolean** before routing begins; read-only **Boolean** while routing is in progress.

expression.**TrackStatus**

expression Required. An expression that returns a [RoutingSlip](#) object.

Example

This example adds a routing slip to the active document, adds two recipients, enables status tracking, and routes the document.

```
ActiveDocument.HasRoutingSlip = True
With ActiveDocument.RoutingSlip
    .AddRecipient Recipient:="James Allard"
    .AddRecipient Recipient:="Rich Andrews"
    .TrackStatus = True
    .Parent.Route
End With
```



↳ [Show All](#)

TrailingCharacter Property

Returns or sets the character inserted after the number for the specified list level.
Read/write [WdTrailingCharacter](#).

WdTrailingCharacter can be one of these WdTrailingCharacter constants.

wdTrailingNone

wdTrailingSpace

wdTrailingTab

expression.**TrailingCharacter**

expression Required. An expression that returns a [ListLevel](#) object.

Example

This example sets the number and text alignment for each level of the sixth outline-numbered list template. The number for each level is followed by a space.

```
r = 0
For Each lev In ListGalleries(wdOutlineNumberGallery) _
    .ListTemplates(6).ListLevels
    lev.Alignment = wdListLevelAlignLeft
    lev.NumberPosition = r
    lev.TextPosition = r
    lev.TrailingCharacter = wdTrailingSpace
    r = r + 18
Next lev
```



Transparency Property

Returns or sets the degree of transparency of the specified fill, shadow, or line as a value between 0.0 (opaque) and 1.0 (clear). Read/write **Single**.

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 of patterned, gradient, picture, or textured fills.

Example

This example sets the shadow of shape three in the active document to semitransparent red. If the shape doesn't already have a shadow, this example adds one to it.

```
With ActiveDocument.Shapes(3).Shadow
    .Visible = True
    .ForeColor.RGB = RGB(255, 0, 0)
    .Transparency = 0.5
End With
```



TransparencyColor Property

Returns or sets the transparent color for the specified picture as a red-green-blue (RGB) value. For this property to take effect, the [TransparentBackground](#) property must be set to **True**. Applies to bitmaps only. Read/write **Long**.

expression.**TransparencyColor**

expression Required. An expression that returns a [PictureFormat](#) object.

Remarks

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 **False**. If your picture has a transparent color and the **Visible** property of the picture's **FillFormat** object is set to **True**, 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 returned by the RGB function as the transparent color for shape one in the active document. For the example to work, shape one must be a bitmap.

```
blueScreen = RGB(0, 0, 255)
With ActiveDocument.Shapes(1)
  With .PictureFormat
    .TransparentBackground = True
    .TransparencyColor = blueScreen
  End With
  .Fill.Visible = False
End With
```



↳ [Show All](#)

TransparentBackground Property

MsoTrue if the parts of the picture that are defined with a transparent color actually appear transparent. Use the [TransparencyColor](#) property to set the transparent color. Applies to bitmaps only. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**TransparentBackground**

expression Required. An expression that returns a [PictureFormat](#) object.

Remarks

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 **False**. If your picture has a transparent color and the **Visible** property of the picture's **FillFormat** object is set to **True**, 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 returned by the RGB function as the transparent color for shape one in the active document. For the example to work, shape one must be a bitmap.

```
blueScreen = RGB(0, 0, 255)
With ActiveDocument.Shapes(1)
  With .PictureFormat
    .TransparentBackground = msoTrue
    .TransparencyColor = blueScreen
  End With
  .Fill.Visible = False
End With
```



TwoInitialCapsAutoAdd Property

-

True if Microsoft Word automatically adds words to the list of AutoCorrect Initial Caps exceptions. A word is added to this list if you delete and then retype the uppercase letter (following the initial uppercase letter) that Word changed to lowercase. Read/write **Boolean**.

expression.**TwoInitialCapsAutoAdd**

expression Required. An expression that returns an [AutoCorrect](#) object.

Example

This example sets Word to automatically add words to the list of AutoCorrect Initial Caps exceptions.

```
AutoCorrect.TwoInitialCapsAutoAdd = True
```



TwoInitialCapsExceptions Property

Returns a [TwoInitialCapsExceptions](#) collection that represents the list of terms containing mixed capitalization that Word won't correct automatically. This list corresponds to the list of AutoCorrect exceptions on the **Initial Caps** tab in the **AutoCorrect Exceptions** dialog box (**AutoCorrect Options** command, **Tools** menu).

expression.TwoInitialCapsExceptions

expression Required. An expression that returns an [AutoCorrect](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example prompts the user to delete or keep each AutoCorrect Initial Caps exception.

```
For Each anEntry In AutoCorrect.TwoInitialCapsExceptions
    response = MsgBox ("Delete entry: " _
        & anEntry.Name, vbYesNoCancel)
    If response = vbYes Then
        anEntry.Delete
    Else
        If response = vbCancel Then End
    End If
Next anEntry
```



↳ [Show All](#)

TwoLinesInOne Property

Returns or sets whether Microsoft Word sets two lines of text in one and specifies the characters that enclose the text, if any. Read/write [WdTwoLinesInOneType](#).

WdTwoLinesInOneType can be one of these WdTwoLinesInOneType constants.

wdTwoLinesInOneCurlyBrackets

wdTwoLinesInOneNone

wdTwoLinesInOneSquareBrackets

wdTwoLinesInOneAngleBrackets

wdTwoLinesInOneNoBrackets

wdTwoLinesInOneParentheses

expression.TwoLinesInOne

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

Remarks

Setting the **TwoLinesInOne** property to **wdTwoLinesInOneNoBrackets** sets two lines of text in one without enclosing the text in any characters. Setting the **TwoLinesInOne** property to **wdTwoLinesInOneNone** restores a line of combined text to two separate lines.

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example formats the current selection as two lines of text in one, enclosed in parentheses.

```
Selection.Range.TwoLinesInOne = _  
    wdTwoLinesInOneParentheses
```



TwoPagesOnOne Property

True if Microsoft Word prints the specified document two pages per sheet.
Read/write **Boolean**.

expression.**TwoPagesOnOne**

expression Required. An expression that returns a [PageSetup](#) object.

Example

This example sets Microsoft Word to print the active document two pages per sheet.

```
ActiveDocument.PageSetup.TwoPagesOnOne = True
```



↳ [Show All](#)

Type Property

▶ [Type property as it applies to the **CalloutFormat** object.](#)

Returns or sets the callout type. Read/write [MsoCalloutType](#).

MsoCalloutType can be one of these MsoCalloutType constants.

msoCalloutFour

msoCalloutOne

msoCalloutTwo

msoCalloutMixed

msoCalloutThree

expression.**Type**

expression Required. An expression that returns a [CalloutFormat](#) object.

▶ [Type property as it applies to the **ColorFormat** object.](#)

Returns or sets the shape color type. Read-only [MsoColorType](#).

MsoColorType can be one of these MsoColorType constants.

msoColorTypeCMYK

msoColorTypeRGB

msoColorTypeCMS

msoColorTypeMixed

msoColorTypeScheme

expression.**Type**

expression Required. An expression that returns [ColorFormat](#) object.

▶ [Type property as it applies to the **Diagram** object.](#)

Returns the diagram type. Read-only [MsoDiagramType](#).

MsoDiagramType can be one of these MsoDiagramType constants.

msoDiagramCycle
msoDiagramMixed
msoDiagramOrgChart
msoDiagramPyramid
msoDiagramRadial
msoDiagramTarget
msoDiagramVenn

expression.Type

expression Required. An expression that returns a [Diagram](#) object.

► [Type property as it applies to the Dialog object.](#)

Returns the type of built-in Microsoft Word dialog box. Read-only [WdWordDialog](#).

WdWordDialog can be one of these WdWordDialog constants.

wdDialogConsistencyChecker
wdDialogConvertObject
wdDialogCreateAutoText
wdDialogDrawAlign
wdDialogEditAutoText
wdDialogEditFind
wdDialogEditGoTo
wdDialogInsertAddCaption
wdDialogInsertBookmark
wdDialogInsertCaption
wdDialogInsertCrossReference
wdDialogInsertDateTime
wdDialogInsertFile
wdDialogInsertFormField

wdDialogInsertIndex
wdDialogInsertMergeField
wdDialogInsertObject
wdDialogInsertPicture
wdDialogInsertSubdocument
wdDialogInsertSymbol
wdDialogInsertTableOfAuthorities
wdDialogInsertTableOfContents
wdDialogInsertTableOfFigures
wdDialogLetterWizard
wdDialogListCommands
wdDialogMailMerge
wdDialogMailMergeCheck
wdDialogMailMergeCreateDataSource
wdDialogMailMergeCreateHeaderSource
wdDialogMailMergeFieldMapping
wdDialogMailMergeFindRecord
wdDialogMailMergeHelper
wdDialogMailMergeInsertAddressBlock
wdDialogMailMergeInsertAsk
wdDialogMailMergeInsertFields
wdDialogMailMergeInsertFillIn
wdDialogMailMergeInsertGreetingLine
wdDialogMailMergeInsertIf
wdDialogMailMergeInsertNextIf
wdDialogMailMergeInsertSet
wdDialogMailMergeInsertSkipIf
wdDialogMailMergeOpenDataSource
wdDialogMailMergeOpenHeaderSource
wdDialogMailMergeQueryOptions
wdDialogMailMergeRecipients
wdDialogMailMergeUseAddressBook
wdDialogMarkCitation

wdDialogMarkIndexEntry
wdDialogMarkTableOfContentsEntry
wdDialogNewToolbar
wdDialogNoteOptions
wdDialogOrganizer
wdDialogPhoneticGuide
wdDialogReviewAfmtRevisions
wdDialogSearch
wdDialogTableAutoFormat
wdDialogTableCellOptions
wdDialogTableColumnWidth
wdDialogTableDeleteCells
wdDialogTableFormatCell
wdDialogTableFormula
wdDialogTableInsertCells
wdDialogTableInsertRow
wdDialogTableInsertTable
wdDialogTableOfCaptionsOptions
wdDialogTableOfContentsOptions
wdDialogTableProperties
wdDialogTableRowHeight
wdDialogTableSort
wdDialogTableSplitCells
wdDialogTableTableOptions
wdDialogTableToText
wdDialogTableWrapping
wdDialogTCSCTranslator
wdDialogTextToTable
wdDialogToolsAcceptRejectChanges
wdDialogToolsAdvancedSettings
wdDialogToolsAutoCorrect
wdDialogToolsAutoManager
wdDialogToolsBulletsNumbers

wdDialogToolsCreateDirectory
wdDialogToolsCreateLabels
wdDialogToolsCustomizeKeyboard
wdDialogToolsCustomizeMenus
wdDialogToolsEnvelopesAndLabels
wdDialogToolsHighlightChanges
wdDialogToolsLanguage
wdDialogToolsMacroRecord
wdDialogToolsMergeDocuments
wdDialogToolsOptionsAutoFormat
wdDialogToolsOptionsBidi
wdDialogToolsOptionsEdit
wdDialogToolsOptionsFuzzy
wdDialogToolsOptionsPrint
wdDialogToolsOptionsSpellingAndGrammar
wdDialogToolsOptionsTypography
wdDialogToolsOptionsView
wdDialogToolsProtectSection
wdDialogToolsSpellingAndGrammar
wdDialogToolsThesaurus
wdDialogToolsWordCount
wdDialogUpdateTOC
wdDialogWebOptions
wdDialogConnect
wdDialogControlRun
wdDialogCopyFile
wdDialogDocumentStatistics
wdDialogDrawSnapToGrid
wdDialogEditCreatePublisher
wdDialogEditFrame
wdDialogEditGoToOld
wdDialogEditLinks
wdDialogEditObject

wdDialogEditPasteSpecial
wdDialogEditPublishOptions
wdDialogEditReplace
wdDialogEditStyle
wdDialogEditSubscribeOptions
wdDialogEditSubscribeTo
wdDialogEditTOACategory
wdDialogEmailOptions
wdDialogFileDocumentLayout
wdDialogFileFind
wdDialogFileMacCustomPageSetupGX
wdDialogFileMacPageSetup
wdDialogFileMacPageSetupGX
wdDialogFileNew
wdDialogFileOpen
wdDialogFilePageSetup
wdDialogFilePrint
wdDialogFilePrintOneCopy
wdDialogFilePrintSetup
wdDialogFileRoutingSlip
wdDialogFileSaveAs
wdDialogFileSaveVersion
wdDialogFileSummaryInfo
wdDialogFileVersions
wdDialogFitText
wdDialogFontSubstitution
wdDialogFormatAddrFonts
wdDialogFormatBordersAndShading
wdDialogFormatBulletsAndNumbering
wdDialogFormatCallout
wdDialogFormatChangeCase
wdDialogFormatColumns
wdDialogFormatDefineStyleBorders

wdDialogFormatDefineStyleFont
wdDialogFormatDefineStyleFrame
wdDialogFormatDefineStyleLang
wdDialogFormatDefineStylePara
wdDialogFormatDefineStyleTabs
wdDialogFormatDrawingObject
wdDialogFormatDropCap
wdDialogFormatEncloseCharacters
wdDialogFormatFont
wdDialogFormatFrame
wdDialogFormatPageNumber
wdDialogFormatParagraph
wdDialogFormatPicture
wdDialogFormatRetAddrFonts
wdDialogFormatSectionLayout
wdDialogFormatStyle
wdDialogFormatStyleGallery
wdDialogFormatStylesCustom
wdDialogFormatTabs
wdDialogFormatTheme
wdDialogFormFieldHelp
wdDialogFormFieldOptions
wdDialogFrameSetProperties
wdDialogHelpAbout
wdDialogHelpWordPerfectHelp
wdDialogHelpWordPerfectHelpOptions
wdDialogHorizontalInVertical
wdDialogIMESetDefault
wdDialogInsertAutoCaption
wdDialogInsertBreak
wdDialogInsertCaptionNumbering
wdDialogInsertDatabase
wdDialogInsertField

wdDialogInsertFootnote
wdDialogInsertHyperlink
wdDialogInsertIndexAndTables
wdDialogInsertNumber
wdDialogInsertPageNumbers
wdDialogToolsAutoCorrectExceptions
wdDialogToolsAutoSummarize
wdDialogToolsCompareDocuments
wdDialogToolsCreateEnvelope
wdDialogToolsCustomize
wdDialogToolsCustomizeMenuBar
wdDialogToolsDictionary
wdDialogToolsHangulHanjaConversion
wdDialogToolsHyphenation
wdDialogToolsMacro
wdDialogToolsManageFields
wdDialogToolsOptions
wdDialogToolsOptionsAutoFormatAsYouType
wdDialogToolsOptionsCompatibility
wdDialogToolsOptionsFileLocations
wdDialogToolsOptionsGeneral
wdDialogToolsOptionsSave
wdDialogToolsOptionsTrackChanges
wdDialogToolsOptionsUserInfo
wdDialogToolsProtectDocument
wdDialogToolsRevisions
wdDialogToolsTemplates
wdDialogToolsUnprotectDocument
wdDialogTwoLinesInOne
wdDialogViewZoom
wdDialogWindowActivate

expression.Type

expression Required. An expression that returns a [Dialog](#) object.

▶ [Type property as it applies to the Dictionary object.](#)

Returns the dictionary type. Read-only [WdDictionaryType](#).

WdDictionaryType can be one of these WdDictionaryType constants.

wdGrammar

wdHangulHanjaConversionCustom

wdSpelling

wdSpellingCustom

wdSpellingMedical

wdHangulHanjaConversion

wdHyphenation

wdSpellingComplete

wdSpellingLegal

wdThesaurus

expression.Type

expression Required. An expression that returns a [Dictionary](#) object.

▶ [Type property as it applies to the Document object.](#)

Returns the document type (template or document). Read-only [WdDocumentType](#).

WdDocumentType can be one of these WdDocumentType constants.

wdTypeDocument

wdTypeTemplate

wdTypeFrameset

expression.Type

expression Required. An expression that returns a [Document](#) object.

▶ [Type property as it applies to the Field, FormField, and MailMergeField](#)

[objects.](#)

Returns the field type. Read-only [WdFieldType](#).

WdFieldType can be one of these WdFieldType constants.

wdFieldFileSize

wdFieldFootnoteRef

wdFieldFormDropDown

wdFieldFormula

wdFieldGoToButton

wdFieldHyperlink

wdFieldImport

wdFieldIncludePicture

wdFieldIndex

wdFieldInfo

wdFieldLastSavedBy

wdFieldListNum

wdFieldMacroButton

wdFieldMergeField

wdFieldMergeRec

wdFieldMergeSeq

wdFieldNext

wdFieldNextIf

wdFieldNoteRef

wdFieldNumChars

wdFieldNumPages

wdFieldNumWords

wdFieldOCX

wdFieldPage

wdFieldPageRef

wdFieldPrint

wdFieldPrintDate

wdFieldPrivate

wdFieldQuote

wdFieldRef
wdFieldRefDoc
wdFieldRevisionNum
wdFieldSaveDate
wdFieldSection
wdFieldSectionPages
wdFieldSequence
wdFieldSet
wdFieldSkipIf
wdFieldStyleRef
wdFieldSubject
wdFieldSubscriber
wdFieldSymbol
wdFieldTemplate
wdFieldTime
wdFieldTitle
wdFieldTOA
wdFieldTOAEntry
wdFieldTOC
wdFieldTOCEntry
wdFieldUserAddress
wdFieldUserInitials
wdFieldUserName
wdFieldAddin
wdFieldAdvance
wdFieldAsk
wdFieldAuthor
wdFieldAutoNum
wdFieldAutoNumLegal
wdFieldAutoNumOutline
wdFieldAutoText
wdFieldAutoTextList
wdFieldBarCode

wdFieldComments
wdFieldCompare
wdFieldCreateDate
wdFieldData
wdFieldDatabase
wdFieldDate
wdFieldDDE
wdFieldDDEAuto
wdFieldDocProperty
wdFieldDocVariable
wdFieldEditTime
wdFieldEmbed
wdFieldEmpty
wdFieldExpression
wdFieldFileName
wdFieldFillIn
wdFieldFormCheckBox
wdFieldFormTextInput
wdFieldGlossary
wdFieldHTMLActiveX
wdFieldIf
wdFieldInclude
wdFieldIncludeText
wdFieldIndexEntry
wdFieldKeyWord
wdFieldLink

expression.**Type**

expression Required. An expression that returns one of the above objects.

► [Type property as it applies to the **FillFormat** object.](#)

Returns the shape fill format type. Read-only [MsoFillType](#).

MsoFillType can be one of these MsoFillType constants.

msoFillGradient

msoFillBackground

msoFillMixed

msoFillPatterned

msoFillPicture

msoFillSolid

msoFillTextured

expression.Type

expression Required. An expression that returns a [FillFormat](#) object.

▶ [Type property as it applies to the Frameset object.](#)

Returns the [Frameset](#) object type. Read-only [WdFramesetType](#).

WdFramesetType can be one of these WdFramesetType constants.

wdFramesetTypeFrame

wdFramesetTypeFrameset

expression.Type

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

▶ [Type property as it applies to the Hyperlink object.](#)

Returns the hyperlink type. Read-only [MsoHyperlinkType](#).

MsoHyperlinkType can be one of these MsoHyperlinkType constants.

msoHyperlinkInlineShape

msoHyperlinkRange

msoHyperlinkShape

expression.Type

expression Required. An expression that returns a [Hyperlink](#) object.

▶ [Type property as it applies to the **Index** object.](#)

Returns or sets the index type. Read/write [WdIndexType](#).

WdIndexType can be one of these WdIndexType constants.

wdIndexRunin

wdIndexIndent

expression.Type

expression Required. An expression that returns an [Index](#) object.

▶ [Type property as it applies to the **InlineShape** object.](#)

Returns the type of inline shape. Read-only [WdInlineShapeType](#).

WdInlineShapeType can be one of these WdInlineShapeType constants.

wdInlineShapeEmbeddedOLEObject

wdInlineShapeHorizontalLine

wdInlineShapeLinkedOLEObject

wdInlineShapeLinkedPicture

wdInlineShapeLinkedPictureHorizontalLine

wdInlineShapeOLEControlObject

wdInlineShapeOWSAnchor

wdInlineShapePicture

wdInlineShapePictureBullet

wdInlineShapePictureHorizontalLine

wdInlineShapeScriptAnchor

expression.Type

expression Required. An expression that returns an [InlineShape](#) object.

▶ [Type property as it applies to the **LinkFormat** object.](#)

Returns the link type. Read-only [WdLinkType](#).

WdLinkType can be one of these WdLinkType constants.

wdLinkTypeText

wdLinkTypeDDE

wdLinkTypeDDEAuto

wdLinkTypeImport

wdLinkTypeInclude

wdLinkTypeOLE

wdLinkTypePicture

wdLinkTypeReference

expression.**Type**

expression Required. An expression that returns a [LinkFormat](#) object.

▶ [Type property as it applies to the MailMergeDataSource object.](#)

Returns the type of mail merge data source. Read-only [WdMailMergeDataSource](#).

WdMailMergeDataSource can be one of these WdMailMergeDataSource constants.

wdMergeInfoFromAccessDDE

wdMergeInfoFromMSQueryDDE

wdMergeInfoFromODSO

wdNoMergeInfo

wdMergeInfoFromExcelDDE

wdMergeInfoFromODBC

wdMergeInfoFromWord

expression.**Type**

expression Required. An expression that returns a [MailMergeDataSource](#) object.

▶ [Type property as it applies to the ProofreadingErrors object.](#)

Returns the type of proofreading error. Read-only [WdProofreadingErrorType](#).

WdProofreadingErrorType can be one of these WdProofreadingErrorType constants.

wdGrammaticalError

wdSpellingError

expression.Type

expression Required. An expression that returns a [ProofreadingErrors](#) object.

▶ [Type property as it applies to the Revision object.](#)

Returns the revision type. Read-only [WdRevisionType](#).

WdRevisionType can be one of these WdRevisionType constants.

wdNoRevision

wdRevisionDelete

wdRevisionInsert

wdRevisionParagraphProperty

wdRevisionReconcile

wdRevisionSectionProperty

wdRevisionStyleDefinition

wdRevisionConflict

wdRevisionDisplayField

wdRevisionParagraphNumber

wdRevisionProperty

wdRevisionReplace

wdRevisionStyle

wdRevisionTableProperty

expression.Type

expression Required. An expression that returns a [Revision](#) object.

▶ [Type property as it applies to the Selection object.](#)

Returns the selection type. Read-only [WdSelectionType](#).

WdSelectionType can be one of these WdSelectionType constants.

wdSelectionBlock

wdSelectionFrame

wdSelectionIP

wdSelectionRow

wdNoSelection

wdSelectionColumn

wdSelectionInlineShape

wdSelectionNormal

wdSelectionShape

expression.**Type**

expression Required. An expression that returns a [Selection](#) object.

► [Type property as it applies to the ShadowFormat object.](#)

Returns or sets the shape shadow type. Read/write [MsoShadowType](#).

MsoShadowType can be one of these MsoShadowType constants.

msoShadow10

msoShadow12

msoShadow14

msoShadow16

msoShadow18

msoShadow2

msoShadow3

msoShadow5

msoShadow7

msoShadow9

msoShadow1

msoShadow11

msoShadow13

msoShadow15

msoShadow17

msoShadow19
msoShadow20
msoShadow4
msoShadow6
msoShadow8
msoShadowMixed

expression.**Type**

expression Required. An expression that returns a [ShadowFormat](#) object.

► [Type property as it applies to the Shape and ShapeRange objects.](#)

Returns the shape type. Read-only [MsoShapeType](#).

MsoShapeType can be one of these MsoShapeType constants.

msoAutoShape
msoCanvas
msoComment
msoFormControl
msoCallout
msoChart
msoEmbeddedOLEObject
msoFreeform
msoGroup
msoLine
msoLinkedOLEObject
msoLinkedPicture
msoMedia
msoOLEControlObject
msoPicture
msoPlaceholder
msoScriptAnchor
msoShapeTypeMixed
msoTable

msoTextBox
msoTextEffect

expression.Type

expression Required. An expression that returns one of the above objects.

▶ [Type property as it applies to the **Style** object.](#)

Returns the style type. Read-only [WdStyleType](#).

WdStyleType can be one of these WdStyleType constants.

wdStyleTypeCharacter
wdStyleTypeList
wdStyleTypeParagraph
wdStyleTypeTable

expression.Type

expression Required. An expression that returns a [Style](#) object.

▶ [Type property as it applies to the **StyleSheet** object.](#)

Returns or sets the style sheet type. Read/write [WdStyleSheetLinkType](#).

WdStyleSheetLinkType can be one of these WdStyleSheetLinkType constants.

wdStyleSheetLinkTypeImported
wdStyleSheetLinkTypeLinked

expression.Type

expression Required. An expression that returns a [StyleSheet](#) object.

▶ [Type property as it applies to the **Template** object.](#)

Returns the template type. Read-only [WdTemplateType](#).

WdTemplateType can be one of these WdTemplateType constants.

wdAttachedTemplate
wdGlobalTemplate
wdNormalTemplate

expression.Type

expression Required. An expression that returns a [Template](#) object.

▶ [Type property as it applies to the TextInput object.](#)

Returns the type of text form field. Read-only [WdTextFormFieldType](#).

WdTextFormFieldType can be one of these WdTextFormFieldType constants.

wdCalculationText
wdCurrentDateText
wdCurrentTimeText
wdDateText
wdNumberText
wdRegularText

expression.Type

expression Required. An expression that returns a [TextInput](#) object.

▶ [Type property as it applies to the View object.](#)

Returns or sets the view type. Read/write [WdViewType](#).

WdViewType can be one of these WdViewType constants.

wdMasterView
wdOutlineView
wdPrintView
wdNormalView
wdPrintPreview
wdWebView

expression.Type

expression Required. An expression that returns a [View](#) object.

Remarks

The **Type** property returns **wdMasterView** for all documents where the current view is an outline or a master document. The current view will never return **wdOutlineView** unless explicitly set first in code.

To check whether the current document is an outline, use the **Type** property and the **Subdocuments** collection's **Count** property. If the **Type** property returns either **wdOutlineView** or **wdMasterView** and the **Count** property returns zero, the document is an outline. For example:

```
Sub VerifyOutlineView()  
    With ActiveWindow.View  
        If .Type = wdOutlineView Or wdMasterView Then  
            If ActiveDocument.Subdocuments.Count = 0 Then  
                .  
                .  
                .  
            End If  
        End If  
    End With  
End Sub
```

► [Type property as it applies to the **Window** object.](#)

Returns the window type. Read-only **WdWindowType**.

WdWindowType can be one of these WdWindowType constants.

wdWindowTemplate

wdWindowDocument

expression.**Type**

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

► [Type property as it applies to the **WrapFormat** object.](#)

Returns the wrap type for the specified shape. Read/write **WdWrapType**.

WdWrapType can be one of these WdWrapType constants.

wdWrapInline
wdWrapNone
wdWrapSquare
wdWrapThrough
wdWrapTight
wdWrapTopBottom

expression.Type

expression Required. An expression that returns a [WrapFormat](#) object.

Example

▶ [As it applies to the **Document** object.](#)

If the active window contains a document, this example redefines the Heading 1 style as centered.

```
If ActiveDocument.ActiveWindow.Type = wdWindowDocument Then
    ActiveDocument.Styles("Heading 1") _
        .ParagraphFormat.Alignment = wdAlignParagraphCenter
End If
```

▶ [As it applies to the **Revision** object.](#)

This example accepts the next revision in the active document if the revision type is inserted text.

```
Set myRev = Selection.NextRevision
If Not (myRev Is Nothing) Then
    If myRev.Type = wdRevisionInsert Then myRev.Accept
End If
```

▶ [As it applies to the **Selection** object.](#)

This example formats the selection as engraved if the selection isn't an insertion point.

```
If Selection.Type <> wdSelectionIP Then
    Selection.Font.Engrave = True
Else
    MsgBox "You need to select some text."
End If
```

▶ [As it applies to the **Style** object.](#)

This example displays a message that indicates the style type of the style named "SubTitle" in the active document.

```
If ActiveDocument.Styles("SubTitle").Type = _
    wdStyleTypeParagraph Then
    MsgBox "Paragraph style"
ElseIf ActiveDocument.Styles("SubTitle").Type = _
```

```
        wdStyleTypeCharacter Then
    MsgBox "Character style"
End If
```

▶ [As it applies to the **View** object.](#)

This example switches the active window to print preview. The **Type** property creates a new print preview window.

```
ActiveDocument.ActiveWindow.View.Type = wdPrintPreview
```



TypeNReplace Property

True for Microsoft Word to replace illegal South Asian characters. Read/write **Boolean**.

expression.**TypeNReplace**

expression Required. An expression that returns an **Options** object.

Example

This example instructs Word to replace illegal South Asian characters.

```
Sub TypeReplace()  
    Application.Options.TypeNReplace = True  
End Sub
```



Underline Property

Returns or sets the type of underline applied to the font or range. Read/write [WdUnderline](#).

WdUnderline can be one of these WdUnderline constants.

wdUnderlineDashHeavy
wdUnderlineDashLongHeavy
wdUnderlineDotDashHeavy
wdUnderlineDotDotDashHeavy
wdUnderlineDottedHeavy
wdUnderlineNone
wdUnderlineThick
wdUnderlineWavyDouble
wdUnderlineWords
wdUnderlineDash
wdUnderlineDashLong
wdUnderlineDotDash
wdUnderlineDotDotDash
wdUnderlineDotted
wdUnderlineDouble
wdUnderlineSingle
wdUnderlineWavy
wdUnderlineWavyHeavy

expression.**Underline**

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

Example

This example applies a double underline to the fourth word in the active document.

```
ActiveDocument.Words(4).Underline = wdUnderlineDouble
```

This example applies a single underline to the selected text.

```
If Selection.Type = wdSelectionNormal Then  
    Selection.Font.Underline = wdUnderlineSingle  
Else  
    MsgBox "You need to select some text."  
End If
```



↳ [Show All](#)

UnderlineColor Property

Returns or sets the 24-bit color of the underline for the specified **Font** object. Can be any valid [WdColor](#) constant or a value returned by Visual Basic's **RGB** function.

WdColor can be one of these WdColor constants.

wdColorGray625

wdColorGray70

wdColorGray80

wdColorGray875

wdColorGray95

wdColorIndigo

wdColorLightBlue

wdColorLightOrange

wdColorLightYellow

wdColorOliveGreen

wdColorPaleBlue

wdColorPlum

wdColorRed

wdColorRose

wdColorSeaGreen

wdColorSkyBlue

wdColorTan

wdColorTeal

wdColorTurquoise

wdColorViolet

wdColorWhite

wdColorYellow

wdColorAqua

wdColorAutomatic
wdColorBlack
wdColorBlue
wdColorBlueGray
wdColorBrightGreen
wdColorBrown
wdColorDarkBlue
wdColorDarkGreen
wdColorDarkRed
wdColorDarkTeal
wdColorDarkYellow
wdColorGold
wdColorGray05
wdColorGray10
wdColorGray125
wdColorGray15
wdColorGray20
wdColorGray25
wdColorGray30
wdColorGray35
wdColorGray375
wdColorGray40
wdColorGray45
wdColorGray50
wdColorGray55
wdColorGray60
wdColorGray65
wdColorGray75
wdColorGray85
wdColorGray90
wdColorGreen
wdColorLavender
wdColorLightGreen

wdColorLightTurquoise

wdColorLime

wdColorOrange

wdColorPink

expression.**UnderlineColor**

expression Required. An expression that returns a [Font](#) object.

Remarks

Setting the UnderlineColor property to **wdColorAutomatic** resets the color of the underline to the color of the text above it.

Example

This example applies a double underline to the third word in the active document and sets the color of the underline to turquoise.

```
With ActiveDocument.Words(3)  
    .Underline = wdUnderlineDouble  
    .Font.UnderlineColor = wdColorTurquoise  
End With
```



Uniform Property

True if all the rows in a table have the same number of columns. Read-only **Boolean**.

expression.**Uniform**

expression Required. An expression that returns a [Table](#) object.

Example

This example creates a table that contains a split cell and then displays a message box that confirms that the table doesn't have the same number of columns for each row.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Selection.Range, 5, 5)
myTable.Cell(3, 3).Split 1, 2
If myTable.Uniform = False Then MsgBox "Table is not uniform"
```

This example determines whether the table that contains the selection has the same number of columns for each row.

```
If Selection.Information(wdWithInTable) = True Then
    MsgBox Selection.Tables(1).Uniform
End If
```



UpdateFieldsAtPrint Property

True if Microsoft Word updates fields automatically before printing a document.
Read/write **Boolean**.

expression.**UpdateFieldsAtPrint**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to update fields automatically before printing, and then it prints the active document.

```
Options.UpdateFieldsAtPrint = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Update fields** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.UpdateFieldsAtPrint
```



UpdateLinksAtOpen Property

True if Microsoft Word automatically updates all embedded OLE links in a document when it's opened. Read/write **Boolean**.

expression.**UpdateLinksAtOpen**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to update embedded OLE links when it opens files.

```
Options.UpdateLinksAtOpen = True
```

This example returns the current status of the **Update automatic links at Open** option on the **General** tab in the **Options** dialog box.

```
temp = Options.UpdateLinksAtOpen
```



UpdateLinksAtPrint Property

True if Microsoft Word updates embedded links to other files before printing a document. Read/write **Boolean**.

expression.**UpdateLinksAtPrint**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to update embedded links automatically before printing, and then it prints the active document.

```
Options.UpdateLinksAtPrint = True  
ActiveDocument.PrintOut
```

This example returns the current status of the **Update links** option on the **Print** tab in the **Options** dialog box (**Tools** menu).

```
temp = Options.UpdateLinksAtPrint
```



UpdateLinksOnSave Property

True if hyperlinks and paths to all supporting files are automatically updated before you save the document as a Web page, ensuring that the links are up-to-date at the time the document is saved. **False** if the links are not updated. The default value is **True**. Read/write **Boolean**.

expression.**UpdateLinksOnSave**

expression Required. An expression that returns a [DefaultWebOptions](#) object.

Remarks

You should set this property to **False** if the location where the document is saved is different from the final location on the Web server and the supporting files are not available at the first location.

Example

This example specifies that links are not updated before the document is saved.

```
Application.DefaultWebOptions.UpdateLinksOnSave = False
```



UpdateStylesOnOpen Property

-

True if the styles in the specified document are updated to match the styles in the attached template each time the document is opened. Read/write **Boolean**.

Example

This example enables the option to update document styles for all open documents and then closes the documents. When any of these documents is reopened, changes to the styles in the attached template will automatically appear in the document.

```
For Each doc In Documents
    doc.UpdateStylesOnOpen = True
    doc.Close SaveChanges:=wdSaveChanges
Next doc
```

This example disables the option to update document styles so that changes made to the styles in the attached template aren't reflected in Report.doc.

```
Documents("Report.doc").UpdateStylesOnOpen = False
```



UpperHeadingLevel Property

-

Returns or sets the starting heading level for a table of contents or table of figures. Corresponds to the starting value used with the \o switch for a Table of Contents (TOC) field. Read/write **Long**.

expression.**UpperHeadingLevel**

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

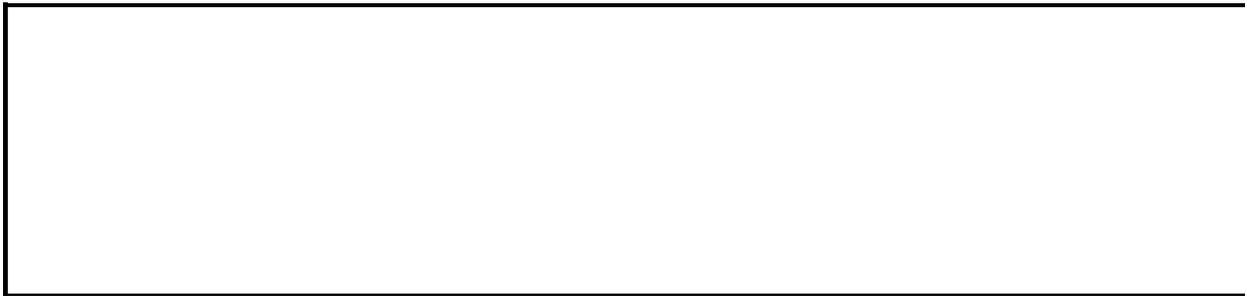
Remarks

Use the [LowerHeadingLevel](#) property to set the ending heading level. For example, to set the TOC field syntax {TOC \o "1-3"}, set the **LowerHeadingLevel** property to 3 and the **UpperHeadingLevel** property to 1.

Example

This example formats the first table of contents in the active document to compile all headings that are formatted with either the Heading 2 or Heading 3 style.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .UseHeadingStyles = True
        .UseFields = False
        .UpperHeadingLevel = 2
        .LowerHeadingLevel = 3
    End With
End If
```



UsableHeight Property

-
Application object: Returns the maximum height (in points) to which you can set the height of a Microsoft Word document window. Read-only **Long**.

Window object: Returns the height (in points) of the active working area in the specified document window. Read-only **Long**. If none of the working area is visible in the document window, **UsableHeight** returns 1. To determine the actual available height, subtract 1 from the **UsableHeight** value.

Example

This example sets the size of the active document window to one quarter of the maximum allowable screen area.

```
With ActiveDocument.ActiveWindow
    .WindowState = wdWindowStateNormal
    .Top = 5
    .Left = 5
    .Height = (Application.UsableHeight*0.5)
    .Width = (Application.UsableWidth*0.5)
End With
```

This example displays the size of the working area in the active document window.

```
With ActiveDocument.ActiveWindow
    MsgBox "Working area height = " _
        & .UsableHeight & vbCrLf _
        & "Working area width = " _
        & .UsableWidth
End With
```



UsableWidth Property

-
Application object: Returns the maximum width (in points) to which you can set the width of a Microsoft Word document window. Read-only **Long**.

Window object: Returns the width (in points) of the active working area in the specified document window. Read-only **Long**. If none of the working area is visible in the document window, **UsableWidth** returns 1. To determine the actual available height, subtract 1 from the **UsableWidth** value.

Example

This example sets the size of the active document window to one quarter of the maximum allowable screen area.

```
With ActiveDocument.ActiveWindow
    .WindowState = wdWindowStateNormal
    .Top = 5
    .Left = 5
    .Height = (Application.UsableHeight*0.5)
    .Width = (Application.UsableWidth*0.5)
End With
```

This example displays the size of the working area in the active document window.

```
With ActiveDocument.ActiveWindow
    MsgBox "Working area height = " _
        & .UsableHeight & vbCrLf _
        & "Working area width = " _
        & .UsableWidth
End With
```



UseCharacterUnit Property

True if Microsoft Word uses characters as the default measurement unit for the current document. Read/write **Boolean**.

expression.**UseCharacterUnit**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets Word to use characters as the default measurement unit.

```
Options.UseCharacterUnit = True
```



UseDiffDiacColor Property

True if you can set the color of diacritics in the current document. Read/write **Boolean**.

expression.**UseDiffDiacColor**

expression Required. An expression that returns an [Options](#) object.

Remarks

For more information on using Word with Asian languages, see [Word features for Asian languages](#).

Example

This example checks the **UseDiffDiacColor** property before setting the color of diacritics in the current selection.

```
If Options.UseDiffDiacColor = True Then _  
    Selection.Font.DiacriticColor = wdColorBlue
```



UseFields Property

-
True if Table of Contents Entry (TC) fields are used to create a table of contents or a table of figures. Read/write **Boolean**.

expression.**UseFields**

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

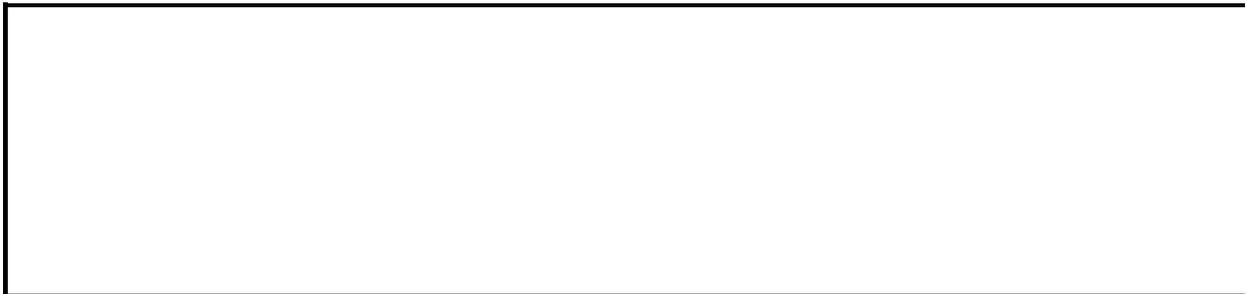
Example

This example formats the first table of contents in the active document to use heading styles instead of TC fields.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .UseFields = False
        .UseHeadingStyles = True
    End With
End If
```

This example adds a table of figures after the selection and formats the table to compile entries with the "B" identifier.

```
Selection.Collapse Direction:=wdCollapseEnd
Set myTOF = ActiveDocument.TablesOfFigures _
    .Add(Range:=Selection.Range)
With myTOF
    .UseFields = True
    .TableId = "B"
    .Caption = ""
End With
```



UseGermanSpellingReform Property

-
True if Microsoft Word uses the German post-reform spelling rules when checking spelling. Read/write **Boolean**.

expression. **UseGermanSpellingReform**

expression Required. An expression that returns an [Options](#) object.

Remarks

This property 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 sets Word to use the post-reform rules for checking spelling in German.

```
Options.UseGermanSpellingReform = True
```



UseHeadingStyles Property

True if built-in heading styles are used to create a table of contents or a table of figures. Read/write **Boolean**.

expression.**UseHeadingStyles**

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

Example

This example formats the first table of contents in the active document to compile entries formatted with the Heading 1, Heading 2, or Heading 3 style.

```
If ActiveDocument.TablesOfContents.Count >= 1 Then
    With ActiveDocument.TablesOfContents(1)
        .UseHeadingStyles = True
        .UseFields = False
        .UpperHeadingLevel = 1
        .LowerHeadingLevel = 3
    End With
End If
```

This example adds a table of figures in place of the selection and then formats the table to compile entries from TC fields.

```
With ActiveDocument.TablesOfFigures.Add(Range:=Selection.Range)
    .UseHeadingStyles = False
    .UseFields = True
End With
```



UseHyperlinks Property

Returns or sets whether entries in a table of contents or a table of figures should be formatted as hyperlinks when publishing to the Web. Read/write **Boolean**.

expression.**UseHyperlinks**

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

Example

This example formats the first table of contents in the document using hyperlinks.

```
ActiveDocument.TableOfContents(1).UseHyperlinks = True
```



UseLongFileNames Property

-

True if long file names are used when you save the document as a Web page.
False if long file names are not used and the DOS file name format (8.3) is used.
The default value is **True**. Read/write **Boolean**.

expression.**UseLongFileNames**

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

Remarks

If you don't use long file names and your document has supporting files, Microsoft Word automatically organizes those files in a separate folder. Otherwise, use the [OrganizeInFolder](#) property to determine whether supporting files are organized in a separate folder.

Example

This example disallows the use of long file names as the global default for the application.

```
Application.DefaultWebOptions.UseLongFileNames = False
```



UserAddress Property

Returns or sets the user's mailing address. Read/write **String**.

expression.**UserAddress**

expression Required. An expression that returns an [Application](#) object.

Remarks

The mailing address is used as a return address on envelopes.

Example

This example sets the user's return address. The **Chr** function is used to return a line feed character.

```
Application.UserAddress = "4200 Third Street NE" & Chr(10) _  
    & "Anytown, WA 98999"
```

This example returns the address found in the **Mailing address** box on the **User Information** tab in the **Options** dialog box (**Tools** menu).

```
Msgbox Application.UserAddress
```



UserControl Property

-

True if the document or application was created or opened by the user. **False** if the document or application was created or opened programmatically from another Microsoft Office application with the **Open** method or the **CreateObject** or **GetObject** method. Read/write **Boolean** for the **Document** object; read-only **Boolean** for the **Application** object.

Remarks

If Word is visible to the user, or if you call the **UserControl** property of a Word **Application** or **Document** object from within a Word code module, this property will always return **True**.

Example

This example displays the status of the **UserControl** property for the active document. This example will only work correctly when run from another Office application with the Word object library loaded.

```
Set wd = New Word.Application
Set wdDoc = _
    wd.Documents.Open("C:\My Documents\doc1.doc")
If wdDoc.UserControl = True Then
    MsgBox "This document was created or opened by the user."
Else
    MsgBox "This document was created programmatically."
End If
```



UserInitials Property

Returns or sets the user's initials, which Microsoft Word uses to construct comment marks. Read/write **String**.

expression.**UserInitials**

expression Required. An expression that returns an [Application](#) object.

Example

This example sets the user's initials.

```
Application.UserInitials = "baa"
```

This example returns the letters found in the **Initials** box on the **User Information** tab in the **Options** dialog box (**Tools** menu).

```
Msgbox Application.UserInitials
```



UserName Property

Returns or sets the user's name, which is used on envelopes and for the Author document property. Read/write **String**.

expression.UserName

expression Required. An expression that returns an [Application](#) object.

Example

This example sets the user's name.

```
Application.UserName = "Andrew Fuller"
```

This example returns the name found in the **Name** box on the **User Information** tab in the **Options** dialog box (**Tools** menu).

```
Msgbox Application.UserName
```



↳ [Show All](#)

UseThemeStyle Property

-

True if new e-mail messages use the character style defined by the default e-mail message [theme](#). If no default e-mail message theme has been specified, this property has no effect. Read/write **Boolean**.

Example

This example sets Microsoft Word to use the Artsy theme as the default theme for new e-mail messages and to use the character style defined in the Artsy theme.

```
Application.EmailOptions.ThemeName = "artsy"  
Application.EmailOptions.UseThemeStyle = True
```



UseThemeStyleOnReply Property

True for Microsoft Word to use a theme when replying to e-mail. Read/write **Boolean**.

expression.**UseThemeStyleOnReply**

expression Required. An expression that returns an [EmailOptions](#) object.

Example

This example tells Word to use a theme when replying to e-mail if Word uses a theme for new messages.

```
Sub NewTheme()  
    With Application.EmailOptions  
        If .UseThemeStyle = True Then  
            .UseThemeStyleOnReply = True  
        End If  
    End With  
End Sub
```



↳ [Show All](#)

Valid Property

-
CheckBox, DropDown, and TextInput objects: **True** if the specified form field object is a valid check box form field. Read-only **Boolean**.

CustomLabel object: **True** if the various properties (for example, **Height, Width, and NumberDown**) for the specified custom label work together to produce a valid mailing label. Read-only **Boolean**.

expression.**Valid**

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

Remarks

For the **CheckBox**, **DropDown**, and **TextInput** objects, use the **Type** property of the **FormField** object to determine the type of form field (**wdFieldFormCheckBox**, **wdFieldFormDropDown**, or **wdFieldFormTextInput**) before applying the **CheckBox**, **DropDown**, or **TextInput** property. This precaution ensures that the **FormField** object is the expected type. If the first form field in the active document is a check box, the following example selects the check box.

```
If ActiveDocument.FormFields(1).Type = wdFieldFormCheckBox Then  
    ActiveDocument.FormFields(1).CheckBox.Valid = True  
End If
```

Example

▶ [As it applies to the **CheckBox** object.](#)

This example adds a text form field at the insertion point. Because myFormField is a text input field and not a check box, the message box displays "False."

```
Selection.Collapse Direction:=wdCollapseStart
Set myFormField = ActiveDocument.FormFields.Add(Range:= _
    Selection.Range, Type:=wdFieldFormTextInput)
MsgBox myFormField.CheckBox.Valid
```

▶ [As it applies to the **TextInput** object.](#)

This example determines whether the first form field in the active document is a text form field. If the **Valid** property is **True**, the contents of the text form field are changed to "Hello."

```
If ActiveDocument.FormFields(1).TextInput.Valid = True Then
    ActiveDocument.FormFields(1).Result = "Hello"
End If
```

▶ [As it applies to the **CustomLabel** object.](#)

If the settings for the custom label named "My Labels" are valid, this example creates a new document of labels using the My Labels settings.

```
addr = "James Allard" & vbCr & "123 Main St." & vbCr _
    & "Seattle, WA 98040"
If Application.MailingLabel.CustomLabels("My Labels") _
    .Valid = True Then
    Application.MailingLabel.CreateNewDocument _
        Name:="My Labels", Address:=addr
End If
```



↳ [Show All](#)

Value Property

▸ [Value property as it applies to the **AutoCorrectEntry**, **AutoTextEntry**, **CustomProperty**, and **Variable** objects.](#)

Returns or sets the value of the AutoCorrect entry, AutoText entry, custom property, or document variable. Read/write **String**.

expression.**Value**

expression Required. An expression that returns one of the above objects.

Remarks

For [AutoCorrectEntry](#) and [AutoTextEntry](#) objects, the **Value** property only returns the first 255 characters of the object's value. Setting the **Value** property to a string longer than 255 characters generates an error.

▶ [Value property as it applies to the DropDown object.](#)

Returns or sets the number of the selected item in a drop-down form field. Read/write **Long**.

expression.**Value**

expression Required. An expression that returns a [DropDown](#) object.

▶ [Value property as it applies to the MailMergeDataField and MappedDataField objects.](#)

Returns the contents of the mail merge data field or mapped data field for the current record. Use the [ActiveRecord](#) property to set the active record in a mail merge data source. Read-only **String**.

expression.**Value**

expression Required. An expression that returns one of the above objects.

▶ [Value property as it applies to the CheckBox object.](#)

True if the check box is selected. Read/write **Boolean**.

expression.**Value**

expression Required. An expression that returns a [CheckBox](#) object.

▶ [Value property as it applies to the ReadabilityStatistic object.](#)

Returns the value of the grammar statistic. Read-only **Long**.

expression.**Value**

expression Required. An expression that returns a [ReadabilityStatistic](#) object.

Example

▶ [As it applies to the **AutoCorrectEntry**, **AutoTextEntry**, **CustomProperty**, and **Variable** objects.](#)

This example adds a document variable to the active document and then displays the value of the new variable.

```
ActiveDocument.Variables.Add Name:="Temp2", Value:="10"  
MsgBox ActiveDocument.Variables("Temp2").Value
```

This example creates an AutoCorrect entry and then displays the value of the new entry.

```
AutoCorrect.Entries.Add Name:="i.e.", Value:="that is"  
MsgBox AutoCorrect.Entries("i.e.").Value
```

▶ [As it applies to the **MailMergeDataField** and **MappedDataField** objects.](#)

This example displays the contents of the active data record in the data source attached to Main.doc.

```
For Each dataF In _  
    Documents("Main.doc").MailMerge.DataSource.DataFields  
    If dataF.Value <> "" Then dRecord = dRecord & _  
        dataF.Value & vbCrLf  
Next dataF  
MsgBox dRecord
```

▶ [As it applies to the **ReadabilityStatistic** object.](#)

This example checks the grammar in the active document and then displays the Flesch reading-ease index.

```
ActiveDocument.CheckGrammar  
MsgBox ActiveDocument.ReadabilityStatistics( _  
    "Flesch Reading Ease").Value
```



Variables Property

-
Returns a [Variables](#) collection that represents the variables stored in the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

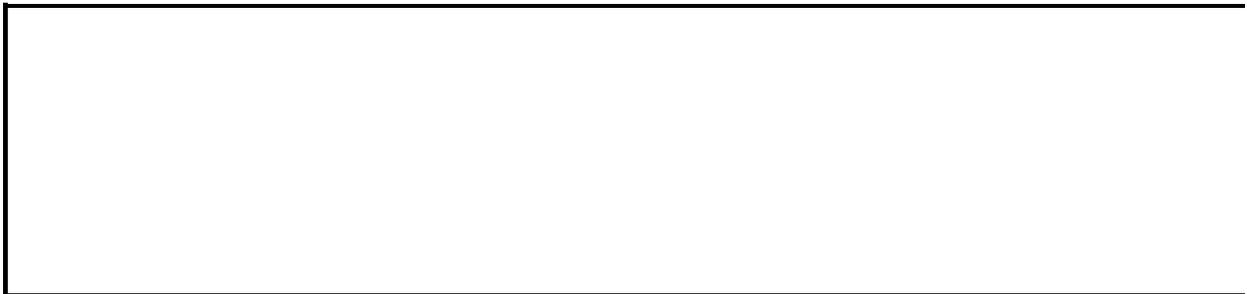
Example

This example adds a document variable named "Value1" to the active document. The example then retrieves the value from the Value1 variable, adds 3 to the value, and displays the results.

```
ActiveDocument.Variables.Add Name:="Value1", Value:="1"  
MsgBox ActiveDocument.Variables("Value1") + 3
```

This example displays the name and value of each document variable in the active document.

```
For Each myVar In ActiveDocument.Variables  
    MsgBox "Name =" & myVar.Name & vbCr & "Value = " & myVar.Value  
Next myVar
```



VBA Signed Property

-
True if the Visual Basic for Applications (VBA) project for the specified document has been digitally signed. Read-only **Boolean**.

Example

This example loads a document called "Temp.doc" and tests to see whether or not it has a digital signature. If there's no digital signature, the example displays a warning message.

```
Documents.Open _  
    FileName:="C:\My Documents\Temp.doc"  
If ActiveDocument.VBASigned = False Then  
    MsgBox "Warning! This document " _  
        & "has not been digitally signed.", _  
        vbCritical, "Digital Signature Warning"  
End If
```



VBE Property

Returns a **VBE** object that represents the Visual Basic Editor.

expression.**VBE**

expression Required. An expression that returns an [Application](#) object.

Example

This example displays the number of references available for the active project.

```
MsgBox "References = " & VBE.ActiveVBProject.References.Count
```



VBProject Property

Returns the **VBProject** object for the specified template or document.

expression.**VBProject**

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

Remarks

Use this property to gain access to code modules and user forms.

To view the **VBProject** object in the object browser, you must select the **Microsoft Visual Basic for Applications Extensibility** check box in the **References** dialog box (**Tools** menu) in the Visual Basic Editor.

Example

This example displays the name of the Visual Basic project for the Normal template.

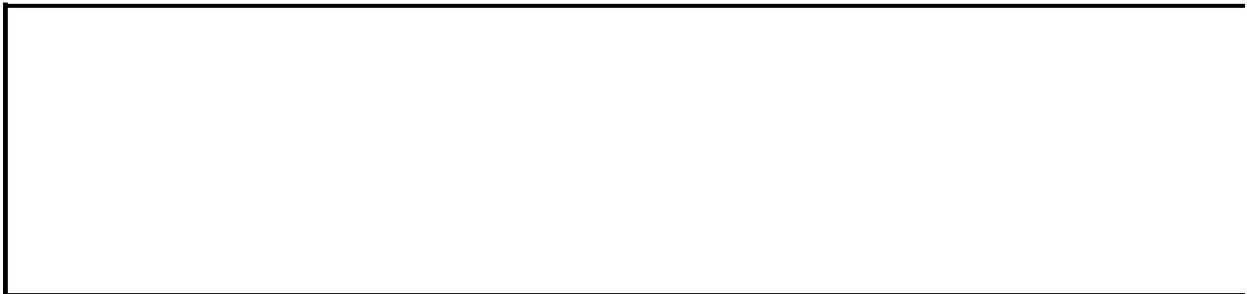
```
Set normProj = NormalTemplate.VBProject
MsgBox normProj.Name
```

This example displays the name of the Visual Basic project for the active document.

```
Set currProj = ActiveDocument.VBProject
MsgBox currProj.Name
```

This example adds a standard code module to the active document and renames it "MyModule."

```
Set newModule = ActiveDocument.VBProject.VBComponents _
    .Add(vbext_ct_StdModule)
NewModule.Name = "MyModule"
```



Version Property

-

Application object: Returns the Microsoft Word version number. Read-only **String**.

System object: Returns the version number of the operating system. Read-only **String**.

Example

This example displays the Word version number in a message box.

```
Msgbox "The version of Word is " & Application.Version
```

This example displays the version number of the operating system in a message box.

```
Msgbox "The system version is " & System.Version
```



Versions Property

-
Returns a [Versions](#) collection that represents all the versions of the specified document. Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the user name and date of the most recent version of the document.

```
If ActiveDocument.Versions.Count >= 1 Then
    Set aVersion = _
        ActiveDocument.Versions(ActiveDocument.Versions.Count)
    MsgBox "Saved by " & aVersion.SavedBy & " on " & aVersion.Date
End If
```

This example saves a version of Contract.doc with a short comment.

```
Documents("Contract.doc").Versions.Save _
    Comment:="Added a single word"
```



Vertical Property

-
True vertically orients text on Asian envelopes and mailing labels. Read/write **Boolean**.

expression.**Vertical**

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

Remark

This property works only with mailing labels or envelopes that are set up for a mail merge and applies only to Asian languages.

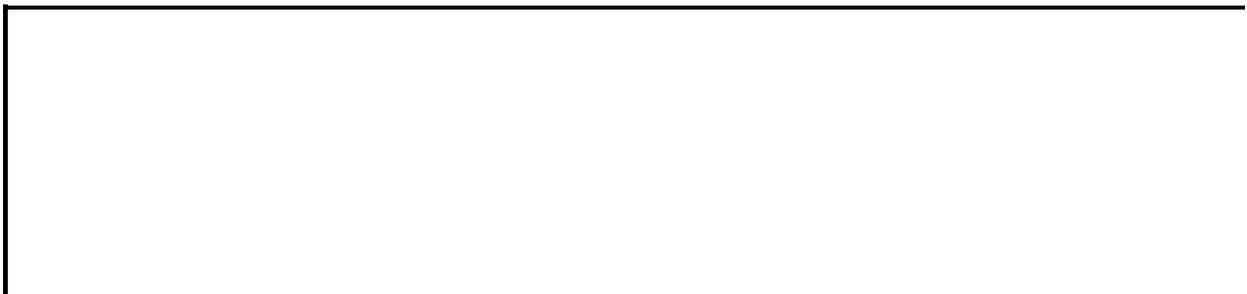
Example

This example determines if the active document is a mail merge mailing label document and if the language setting is Japanese, and if so, sets the mailing label's orientation to vertical.

```
Sub VerticalLabel()  
    If ActiveDocument.MailMerge.MainDocumentType = wdMailingLabels And  
        Application.Language = msoLanguageIDJapanese Then  
        Application.MailingLabel.Vertical = True  
    End If  
End Sub
```

This example determines if the active document is a mail merge envelope document and if the language setting is Chinese, and if so, sets the envelope's orientation to vertical and updates the current document.

```
Sub VerticalEnvelope()  
    If ActiveDocument.MailMerge.MainDocumentType = wdEnvelopes And  
        Application.Language = msoLanguageIDChineseHongKong Then  
        With ThisDocument.Envelope  
            Vertical = True  
            .UpdateDocument  
        End With  
    End If  
End Sub
```



↳ [Show All](#)

VerticalAlignment Property

▶ [VerticalAlignment property as it applies to the Cell and Cells objects.](#)

Returns or sets the vertical alignment of text in one or more cells of a table. Read/write [WdCellVerticalAlignment](#).

WdCellVerticalAlignment can be one of these WdCellVerticalAlignment constants.

wdCellAlignVerticalBottom

wdCellAlignVerticalCenter

wdCellAlignVerticalTop

expression.**VerticalAlignment**

expression Required. An expression that returns one of the above objects.

▶ [VerticalAlignment property as it applies to the PageSetup object.](#)

Returns or sets the vertical alignment of text on each page in a document or section. Read/write [WdVerticalAlignment](#).

WdVerticalAlignment can be one of these WdVerticalAlignment constants.

wdAlignVerticalBottom

wdAlignVerticalCenter

wdAlignVerticalJustify

wdAlignVerticalTop

expression.**VerticalAlignment**

expression Required. An expression that returns a [PageSetup](#) object.

Example

▶ [As it applies to the **Cell** and **Cells** objects.](#)

This example creates a 3x3 table in a new document and assigns a sequential cell number to each cell in the table. The example then sets the height of the first row to 20 points and vertically aligns the text at the top of the cells.

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Selection.Range, 3, 3)
i = 1
For Each c In myTable.Range.Cells
    c.Range.InsertAfter "Cell " & i
    i = i + 1
Next
With myTable.Rows(1)
    .Height = 20
    .Cells.VerticalAlignment = wdAlignVerticalTop
End With
```

▶ [As it applies to the **PageSetup** object.](#)

This example changes the vertical alignment of the first document so that the text is centered between the top and bottom margins.

```
Documents(1).PageSetup.VerticalAlignment = wdAlignVerticalCenter
```

This example creates a new document and then inserts the same paragraph 10 times. The vertical alignment of the new document is then set so that the 10 paragraphs are equally spaced (justified) between the top and bottom margins.

```
Set myDoc = Documents.Add
With myDoc.Content
    For i = 1 to 9
        .InsertAfter "This is a sentence."
        .InsertParagraphAfter
    Next i
    .InsertAfter "This is a sentence."
End With
myDoc.PageSetup.VerticalAlignment = wdAlignVerticalJustify
```



VerticalDistanceFromText Property

Returns or sets the vertical distance (in points) between a frame and the surrounding text. Read/write **Single**.

expression.**VerticalDistanceFromText**

expression Required. An expression that returns a [Frame](#) object.

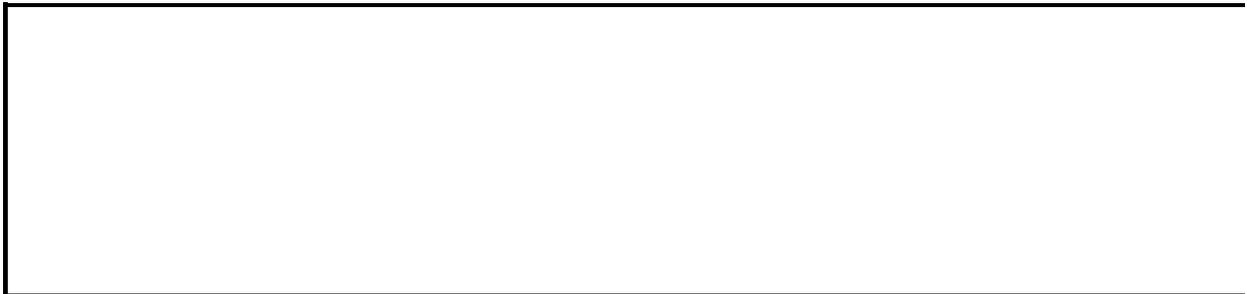
Example

This example sets the vertical distance between the selected frame and the surrounding text to 12 points.

```
If Selection.Frames.Count = 1 Then
    Selection.Frames(1).VerticalDistanceFromText = 12
End If
```

This example adds a frame around the selection and sets several properties of the frame.

```
Set aFrame = ActiveDocument.Frames.Add(Range:=Selection.Range)
With aFrame
    .HorizontalDistanceFromText = InchesToPoints(0.13)
    .VerticalDistanceFromText = InchesToPoints(0.13)
    .HeightRule = wdFrameAuto
    .WidthRule = wdFrameAuto
End With
```



↳ [Show All](#)

VerticalFlip Property

-
True if the specified shape is flipped around the vertical axis. Read-only [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**VerticalFlip**

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

Example

This example restores each shape on myDocument to its original state if it's been flipped horizontally or vertically.

```
For Each s In ActiveDocument.Shapes
    If s.HorizontalFlip Then s.Flip msoFlipHorizontal
    If s.VerticalFlip Then s.Flip msoFlipVertical
Next
```



VerticalPercentScrolled Property

Returns or sets the vertical scroll position as a percentage of the document length. Read/write **Long**.

expression.**VerticalPercentScrolled**

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

Example

This example displays the percentage that the active window is scrolled vertically.

```
MsgBox ActiveDocument.ActiveWindow.VerticalPercentScrolled & "%"
```

This example scrolls the active window vertically by 10 percent.

```
Set aWindow = ActiveDocument.ActiveWindow  
aWindow.VerticalPercentScrolled = _  
    aWindow.VerticalPercentScrolled + 10
```

This example vertically scrolls the active pane of the window for Document1 to the end.

```
With Windows("Document1")  
    .Activate  
    .ActivePane.VerticalPercentScrolled = 100  
End With
```



VerticalPitch Property

Returns or sets the vertical distance between the top of one mailing label and the top of the next mailing label. Read/write **Single**.

expression.**VerticalPitch**

expression Required. An expression that returns a [CustomLabel](#) object.

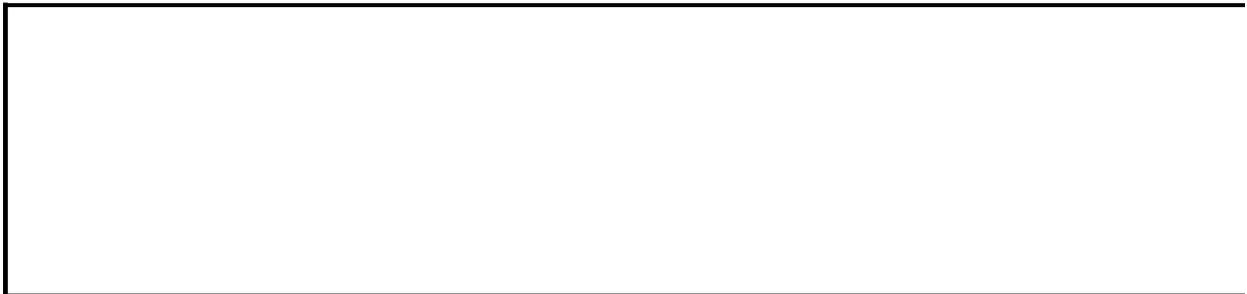
Remarks

If this property is changed to a value that isn't valid for the specified mailing label layout, an error occurs.

Example

This example creates a custom label named "VisitorPass" and defines its layout. The distance between the top edge of one label to the top edge of the next label is 2.17 inches.

```
Set myLabel = Application.MailingLabel.CustomLabels _  
    .Add(Name:="VisitorPass", DotMatrix:=False)  
With myLabel  
    .Height = InchesToPoints(2.17)  
    .HorizontalPitch = InchesToPoints(3.5)  
    .NumberAcross = 2  
    .NumberDown = 4  
    .PageSize = wdCustomLabelLetter  
    .SideMargin = InchesToPoints(0.75)  
    .TopMargin = InchesToPoints(0.17)  
    .VerticalPitch = InchesToPoints(2.17)  
    .Width = InchesToPoints(3.5)  
End With
```



VerticalPosition Property

Returns or sets the vertical distance between the edge of the frame (for the [Frame](#) object) or the rows (for the [Rows](#) object) and the item specified by the [RelativeVerticalPosition](#) property. Can be a number that indicates a measurement in points, or can be any valid **WdFramePosition** constant. For a list of valid constants, consult the Microsoft Visual Basic Object Browser. Read/write **Single**.

expression.**VerticalPosition**

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

Example

This example vertically aligns the first frame in the active document with the top of the page.

```
Set myFrame = ActiveDocument.Frames(1)
With myFrame
    .RelativeVerticalPosition = wdRelativeVerticalPositionPage
    .VerticalPosition = wdFrameTop
End With
```

This example adds a frame around the first shape in the active document and positions the frame 1 inch from the top margin.

```
If ActiveDocument.Shapes.Count >= 1 Then
    ActiveDocument.Shapes(1).Select
    Set aFrame = ActiveDocument.Frames.Add(Range:=Selection.Range)
    With aFrame
        .RelativeVerticalPosition = _
            wdRelativeVerticalPositionMargin
        .VerticalPosition = InchesToPoints(1)
    End With
End If
```

This example vertically aligns the first table in the active document with the top of the page.

```
Set myTable = ActiveDocument.Tables(1).Rows
With myTable
    .RelativeVerticalPosition = wdRelativeVerticalPositionPage
    .VerticalPosition = wdTableTop
End With
```



VerticalResolution Property

Returns the vertical screen resolution in pixels. Read-only **Long**.

expression.**VerticalResolution**

expression Required. An expression that returns a [System](#) object.

Example

This example displays the current screen resolution (for example, "1024 x 768").

```
horz = System.HorizontalResolution  
vert = System.VerticalResolution  
MsgBox "Resolution = " & horz & " x " & vert
```



↳ [Show All](#)

Vertices Property

-

Returns the coordinates of the specified freeform drawing's vertices (and control points for Bézier curves) as a series of [coordinate pairs](#). You can use the array returned by this property as an argument for the [AddCurve](#) or [AddPolyLine](#) method. Read-only **Variant**.

expression.**Vertices**

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

Remarks

The following table shows how the **Vertices** property associates values in the array `vertArray()` with the coordinates of a triangle's vertices.

vertArray element	Contains
<code>vertArray(1, 1)</code>	The horizontal distance from the first vertex to the left side of the document.
<code>vertArray(1, 2)</code>	The vertical distance from the first vertex to the top of the document.
<code>vertArray(2, 1)</code>	The horizontal distance from the second vertex to the left side of the document.
<code>vertArray(2, 2)</code>	The vertical distance from the second vertex to the top of the document.
<code>vertArray(3, 1)</code>	The horizontal distance from the third vertex to the left side of the document.
<code>vertArray(3, 2)</code>	The vertical distance from the third vertex to the top of the document.

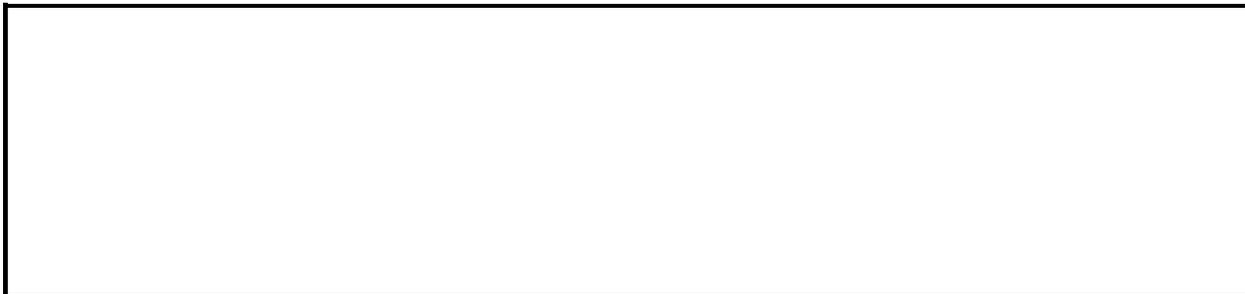
Example

This example assigns the vertex coordinates for shape one in the active document to an array variable and displays the coordinates for the first vertex. Shape one must be a freeform drawing.

```
With ActiveDocument.Shapes(1)
    vertArray = .Vertices
    x1 = vertArray(1, 1)
    y1 = vertArray(1, 2)
    MsgBox "First vertex coordinates: " & x1 & ", " & y1
End With
```

This example creates a curve that has the same geometric description as shape one in the active document. This example assumes that the first shape is a Bézier curve containing $3n+1$ vertices, where n is the number of curve segments.

```
With ActiveDocument.Shapes
    .AddCurve .Item(1).Vertices, Selection.Range
End With
```



View Property

Returns a [View](#) object that represents the view for the specified window or pane.

expression.**View**

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

Example

This example switches the active window to full-screen view.

```
ActiveDocument.ActiveWindow.View.FullScreen = True
```

This example shows all nonprinting characters for panes associated with the first window in the [Windows](#) collection.

```
For Each myPane In Windows(1).Panes  
    myPane.View.ShowAll = True  
Next myPane
```

This example sets view options for each window in the **Windows** collection.

```
For Each myWindow In Windows  
    With myWindow.View  
        .ShowTabs = True  
        .ShowParagraphs = True  
        .Type = wdNormalView  
    End With  
Next myWindow
```



ViewMailMergeFieldCodes Property

True if merge field names are displayed in a mail merge main document. **False** if information from the current data record is displayed. Read/write **Long**.

expression.**ViewMailMergeFieldCodes**

expression Required. An expression that returns a [MailMerge](#) object.

Remarks

If the active document isn't a mail merge main document, this property causes an error. To view merge field names or their results, set the [ShowFieldCodes](#) property to **False**.

Example

This example displays the mail merge fields in Main.doc.

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
With Documents("Main.doc")
    .Activate
    .MailMerge.ViewMailMergeFieldCodes = True
End With
```

If the active document is set up for a mail merge operation, this example displays the current data record information in the main document.

```
ActiveDocument.ActiveWindow.View.ShowFieldCodes = False
Set myMerge = ActiveDocument.MailMerge
If myMerge.State = wdMainAndSourceAndHeader Or _
    myMerge.State = wdMainAndDataSource Then
    myMerge.ViewMailMergeFieldCodes = False
End If
```



↳ [Show All](#)

ViewType Property

Returns or sets the view for the [TextRetrievalMode](#) object. Read/write [WdViewType](#).

WdViewType can be one of these WdViewType constants.

wdMasterView

wdNormalView

wdOutlineView

wdPrintPreview

wdPrintView

wdWebView

expression.**ViewType**

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

Remarks

Changing the view for the **TextRetrievalMode** object doesn't change the display of a document on the screen. Instead, it determines which characters in the document will be included when a range is retrieved.

Example

This example sets the view for text retrieval to outline view and then displays the contents of the active document in a dialog box. Note that only the text displayed in outline view is retrieved.

```
Set myText = ActiveDocument.Content  
myText.TextRetrievalMode.ViewType = wdOutlineView  
Msgbox myText
```



↳ [Show All](#)

Visible Property

▶ [Visible property as it applies to the **FillFormat**, **LineFormat**, **ShadowFormat**, **Shape**, **ShapeRange**, and **ThreeDFormat** objects.](#)

True if the specified object, or the formatting applied to it, is visible. Read/write [MsoTriState](#).

MsoTriState can be one of these MsoTriState constants.

msoCTrue

msoFalse

msoTriStateMixed

msoTriStateToggle

msoTrue

expression.**Visible**

expression Required. An expression that returns one of the above objects.

▶ [Visible property as it applies to the **Application**, **Border**, **Reviewer**, **Task**, **TaskPane**, and **Window** objects.](#)

True if the specified object is visible. Read/write **Boolean**.

expression.**Visible**

expression Required. An expression that returns one of the above objects.

Remarks

For any object, some methods and properties may be unavailable if the **Visible** property is **False**.

Example

▶ [As it applies to the **Application** object.](#)

This example hides Microsoft Word.

```
Application.Visible = False
```

▶ [As it applies to the **Task** object.](#)

This example hides the Calculator, if it's running. If it's not running, a message is displayed.

```
If Tasks.Exists("Calculator") Then
    Tasks("Calculator").Visible = False
Else
    MsgBox "Calculator is not running."
End If
```

▶ [As it applies to the **Border** object.](#)

This example creates a table in the active document and removes the default borders from the table.

```
Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _
    NumRows:=12, NumColumns:=5)
For Each aBorder In myTable.Borders
    aBorder.Visible = False
Next aBorder
```

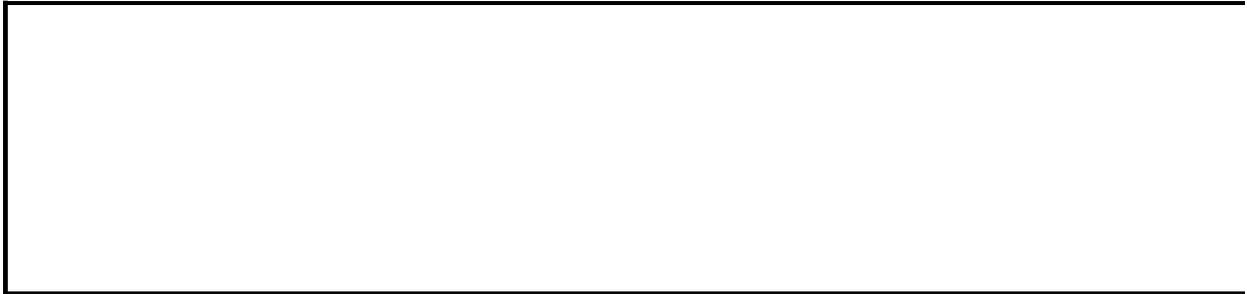
▶ [As it applies to the **Shape** object.](#)

This example hides the shadow formatting for the first shape in the active document.

```
ActiveDocument.Shapes(1).Shadow.Visible = False
```

This example creates a new document and then adds text and a rectangle to it. The example also sets Word to hide the rectangle while the document is being printed and then to make it visible again after printing is completed.

```
Set myDoc = Documents.Add
Selection.TypeText Text:="This is some sample text."
With myDoc
    .Shapes.AddShape msoShapeRectangle, 200, 70, 150, 60
    .Shapes(1).Visible = False
    .PrintOut
    .Shapes(1).Visible = True
End With
```



↳ [Show All](#)

VisualSelection Property

Returns or sets the selection behavior based on visual cursor movement in a right-to-left language document. Read/write [WdVisualSelection](#).

WdVisualSelection can be one of these WdVisualSelection constants.

wdVisualSelectionBlock All selected lines are the same width.

wdVisualSelectionContinuous The selection wraps from line to line.

expression.**VisualSelection**

expression Required. An expression that returns an [Options](#) object.

Remarks

The [CursorMovement](#) property must be set to **wdCursorMovementVisual** in order to use this property.

For more information on using Word with right-to-left languages, see [Word features for right-to-left languages](#).

Example

This example sets the selection behavior so that the selection wraps from line to line.

```
If Options.CursorMovement = wdCursorMovementVisual Then _  
    Options.VisualSelection = wdVisualSelectionContinuous
```



WarnBeforeSavingPrintingSendingMarkup Property

True for Microsoft Word to display a warning when saving, printing, or sending as e-mail a document containing comments or tracked changes. Read/write **Boolean**.

expression.WarnBeforeSavingPrintingSendingMarkup

expression Required. An expression that returns a [Options](#) object.

Example

This example prints the active document but allows the user to abort if the document contains tracked changes or comments.

```
Sub SaferPrint
    Dim blnOldState as Boolean

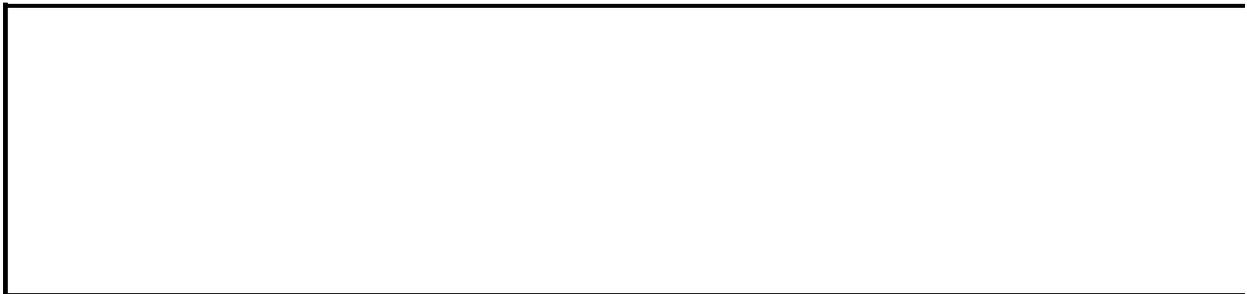
    'Save old state in variable
    blnOldState = Application.Options.WarnBeforeSavingPrintingSendin

    'Turn on warning
    Application.Options.WarnBeforeSavingPrintingSendingMarkup = True

    'Print document
    ActiveDocument.PrintOut

    'Restore original warning state
    Application.Options.WarnBeforeSavingPrintingSendingMarkup = blnC

EndSub
```



WebOptions Property

-

Returns the [WebOptions](#) object, which contains document-level attributes used by Microsoft Word when you save a document as a Web page or open a Web page. Read-only.

Example

This example specifies that cascading style sheets and Western document encoding be used when items in the active document are saved to a Web page.

```
Set objW0 = ActiveDocument.WebOptions  
objW0.RelyOnCSS = True  
objW0.Encoding = msoEncodingWestern
```



Weight Property

Returns or sets the thickness of the specified line in points. Read/write **Single**.

expression.**Weight**

expression Required. An expression that returns a [LineFormat](#) object.

Example

This example adds a green dashed line two points thick to the active document.

```
With ActiveDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(0, 255, 255)
    .Weight = 2
End With
```



WidowControl Property

-

True if the first and last lines in the specified paragraph remain on the same page as the rest of the paragraph when Word repaginates the document. Can be **True**, **False** or **wdUndefined**. Read/write **Long**.

Example

This example formats the paragraphs in the active document so that the first or last line in a paragraph can appear by itself at the top or bottom of a page.

```
ActiveDocument.Paragraphs.WidowControl = False
```



Width Property

-
Frameset object: Returns or sets the width of the specified **Frameset** object.
Read/write **Long**. The [WidthType](#) property determines the type of unit in which this value is expressed.

All other objects: Returns or sets the width of the specified object, in points.
Read/write **Long**.

Example

This example creates a 5x5 table in a new document and then sets the width of the first cell to 1.5 inches.

```
Set newDoc = Documents.Add
Set myTable = _
    newDoc.Tables.Add(Range:=Selection.Range, NumRows:=5, _
        NumColumns:=5)
myTable.Cell(1, 1).Width = InchesToPoints(1.5)
```

This example returns the width (in inches) of the cell that contains the insertion point.

```
If Selection.Information(wdWithInTable) = True Then
    MsgBox PointsToInches(Selection.Cells(1).Width)
End If
```

This example formats the section that includes the selection as three columns. The **For Each...Next** loop is used to display the width of each column in the **TextColumns** collection.

```
Selection.PageSetup.TextColumns.SetCount NumColumns:=3
For Each acol In Selection.PageSetup.TextColumns
    MsgBox "Width= " & PointsToInches(acol.Width)
Next acol
```

This example sets the width and height of the Microsoft Word application window.

```
With Application
    .WindowState = wdWindowStateNormal
    .Width = 500
    .Height = 400
End With
```

This example sets the width of the specified **Frameset** object to 25% of the window width.

```
With ActiveWindow.ActivePane.Frameset
    .WidthType = wdFramesetSizeTypePercent
    .Width = 25
End With
```

End With

--

↳ [Show All](#)

WidthRule Property

Returns or sets the rule used to determine the width of a frame. Read/write [WdFrameSizeRule](#).

WdFrameSizeRule can be one of these WdFrameSizeRule constants.

wdFrameAtLeast Sets the width to a value equal to or greater than the value specified by the [Width](#) property.

wdFrameAuto Sets the width according to the width of the item in the frame.

wdFrameExact Sets the width to an exact value specified by the [Width](#) property.

expression.**WidthRule**

expression Required. An expression that returns a [Frame](#) object.

Example

This example sets the width of the last frame in the active document to exactly 72 points (1 inch).

```
If ActiveDocument.Frames.Count >= 1 Then
    With ActiveDocument.Frames(ActiveDocument.Frames.Count)
        .WidthRule = wdFrameExact
        .Width = 72
    End With
End If
```



↳ [Show All](#)

WidthType Property

▶ [WidthType property as it applies to the Frameset object.](#)

Returns or sets the width type for the specified [Frameset](#) object. Read/write [WdFramesetSizeType](#).

WdFramesetSizeType can be one of these WdFramesetSizeType constants.

wdFramesetSizeTypeFixed Microsoft Word interprets the width of the specified frame as a fixed value (in points).

wdFramesetSizeTypePercent Word interprets the width of the specified frame as a percentage of the screen width.

wdFramesetSizeTypeRelative Word interprets the width of the specified frame as relative to the width of other frames on the frames page.

expression.**WidthType**

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

▶ [WidthType property as it applies to the HorizontalLineFormat object.](#)

Returns or sets the width type for the specified [HorizontalLineFormat](#) object. Read/write [WdHorizontalLineWidthType](#).

WdHorizontalLineWidthType can be one of these WdHorizontalLineWidthType constants.

wdHorizontalLineFixedWidth Microsoft Word interprets the width (length) of the specified horizontal line as a fixed value (in points). This is the default value for horizontal lines added with the [AddHorizontalLine](#) method. Setting the [Width](#) property for the [InlineShape](#) object associated with a horizontal line sets the **WidthType** property to this value.

wdHorizontalLinePercentWidth Word interprets the width (length) of the specified horizontal line as a percentage of the screen width. This is the default value for horizontal lines added with the [AddHorizontalLineStandard](#)

method. Setting the [PercentWidth](#) property on a horizontal line sets the **WidthType** property to this value.

expression.**WidthType**

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

Example

▶ [As it applies to the **Frameset** object.](#)

This example sets the width of the first **Frameset** object in the active document to 25% of the window width.

```
With ActiveDocument.ActiveWindow.Panes(1).Frameset
    .WidthType = wdFramesetSizeTypePercent
    .Width = 25
End With
```

▶ [As it applies to the **HorizontalLineFormat** object.](#)

This example adds horizontal lines to the active document and compares their width types.

```
Dim temp As InlineShape
Set temp = _
    ActiveDocument.InlineShapes.AddHorizontalLineStandard
MsgBox "AddHorizontalLineStandard - WidthType = " _
    & temp.HorizontalLineFormat.WidthType
Set temp = _
    ActiveDocument.InlineShapes.AddHorizontalLine _
    ("C:\My Documents\ArtsyRule.gif")
MsgBox "AddHorizontalLine - WidthType = " _
    & temp.HorizontalLineFormat.WidthType
```



WindowNumber Property

Returns the window number of the document displayed in the specified window. For example, if the caption of the window is "Sales.doc:2", this property returns the number 2. Read-only **Long**.

expression.**WindowNumber**

expression Required. An expression that returns a [Window](#) object.

Remarks

Use the [Index](#) property to return the number of the specified window in the [Windows](#) collection.

Example

This example retrieves the window number of the active window, opens a new window, and then activates the original window.

```
Sub WinNum()  
    Dim lwindowNum As Long  
  
    lwindowNum = ActiveDocument.ActiveWindow.WindowNumber  
    NewWindow  
    ActiveDocument.Windows(lwindowNum).Activate  
End Sub
```



Windows Property

-
Application object: Returns a [Windows](#) collection that represents all document windows. The collection corresponds to the window names that appear at the bottom of the **Window** menu. Read-only.

Document object: Returns a [Windows](#) collection that represents all windows for the specified document (for example, Sales.doc:1 and Sales.doc:2). Read-only.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

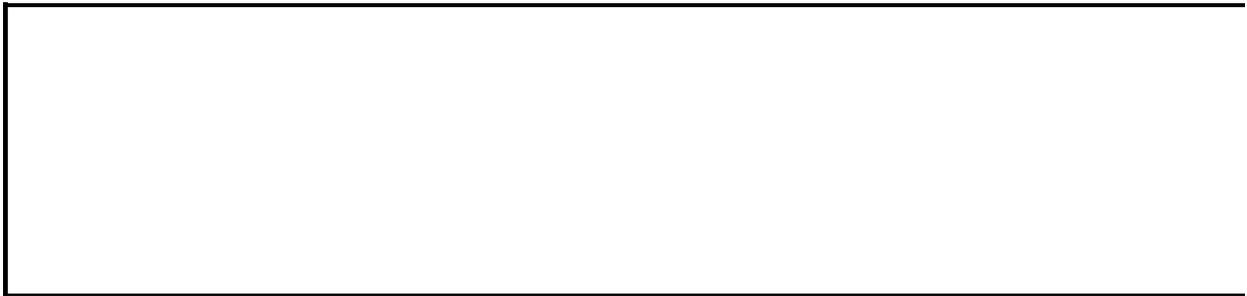
Example

This example displays the number of windows for the active document, both before and after the **NewWindow** method is run.

```
MsgBox Prompt:= ActiveDocument.Windows.Count & " window(s)", _  
    Title:= ActiveDocument.Name  
ActiveDocument.ActiveWindow.NewWindow  
MsgBox Prompt:= ActiveDocument.Windows.Count & " windows", _  
    Title:= ActiveDocument.Name
```

This example arranges all open windows so that they don't overlap.

```
Windows.Arrange ArrangeStyle:=wdTiled
```



WindowState Property

Returns or sets the state of the specified document window or task window.
Read/write [WdWindowState](#).

WdWindowState can be one of these WdWindowState constants.

wdWindowStateMaximize

wdWindowStateNormal

wdWindowStateMinimize

expression.**WindowState**

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

Remarks

The **wdWindowStateNormal** constant indicates a window that's not maximized or minimized. The state of an inactive window cannot be set. Use the [Activate](#) method to activate a window prior to setting the window state.

Example

This example maximizes the active window if it's not maximized or minimized.

```
If ActiveDocument.ActiveWindow _  
    .WindowState = wdWindowStateNormal Then _  
    ActiveDocument.ActiveWindow.WindowState = wdWindowStateMaximize
```

This example minimizes the Microsoft Excel application window.

```
For Each myTask In Tasks  
    If InStr(myTask.Name, "Microsoft Excel") > 0 Then  
        myTask.Activate  
        myTask.WindowState = wdWindowStateMinimize  
    End If  
Next myTask
```



WizardState Property

-

Returns or sets a **Long** indicating the current Mail Merge Wizard step for a document. The **WizardState** method 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 one of the objects in the Applies To list.

Example

This example checks if the Mail Merge Wizard is already displayed in the active document and if it is, moves to the Mail Merge Wizard's sixth step and removes the fifth step from the Wizard.

```
Sub ShowMergeWizard()  
    With ActiveDocument.MailMerge  
        If .WizardState > 0 Then  
            .ShowWizard InitialState:=6, ShowPreviewStep:=False  
        End If  
    End With  
End Sub
```



Word Property

-
Returns the word or phrase that was looked up by the thesaurus. Read-only **String**.

expression.**Word**

expression Required. An expression that returns a [SynonymInfo](#) object.

Remarks

The thesaurus will sometimes look up a shortened version of the string or range used to return the [SynonymInfo](#) object. The **Word** property allows you to see the exact string that was used.

Example

This example returns a list of synonyms for the first meaning of the third word in the active document.

```
Sub Syn()  
    Dim mySynObj As Object  
    Dim SList As Variant  
    Dim i As Variant  
  
    Set mySynObj = ActiveDocument.Words(3).SynonymInfo  
    SList = mySynObj.SynonymList(1)  
    For i = 1 To UBound(SList)  
        MsgBox "A synonym for " & mySynObj.Word _  
            & " is " & SList(i)  
    Next i  
End Sub
```

This example checks to make sure that the word or phrase that was looked up isn't empty. If it's not, the example returns a list of synonyms for the first meaning of the word or phrase.

```
Sub SelectWord()  
    Dim mySynObj As Object  
    Dim SList As Variant  
    Dim i As Variant  
  
    Set mySynObj = Selection.Range.SynonymInfo  
    If mySynObj.Word = "" Then  
        MsgBox "Please select a word or phrase"  
    Else  
        SList = mySynObj.SynonymList(1)  
        For i = 1 To UBound(SList)  
            MsgBox "A synonym for " & mySynObj.Word _  
                & " is " & SList(i)  
        Next i  
    End If  
End Sub
```



WordBasic Property

-

Returns an Automation object (Word.Basic) that includes methods for all the WordBasic statements and functions available in Word version 6.0 and Word for Windows 95. Read-only.

Remarks

In Word 2000 and later, when you open a Word version 6.0 or Word for Windows 95 template that contains WordBasic macros, the macros are automatically converted to Visual Basic modules. Each WordBasic statement and function in the macro is converted to the corresponding Word.Basic method.

For information about WordBasic statements and functions, see WordBasic Help in Word version 6.0 or Word for Windows 95.

For information about converting WordBasic to Visual Basic, see [Visual Basic Equivalents for WordBasic Commands](#).

Example

This example uses the Word.Basic object to create a new document in Word version 6.0 or Word for Windows 95 and insert the available font names. Each font name is formatted in its corresponding font.

```
With WordBasic
  .FileNewDefault
  For aCount = 1 To .CountFonts()
    .Font .[Font$](aCount)
    .Insert .[Font$](aCount)
    .InsertPara
  Next
End With
```



Words Property

-
Returns a [Words](#) collection that represents all the words in a range, selection, or document. Read-only.

Note Punctuation and paragraph marks in a document are included in the **Words** collection.

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example displays the number of words in the selection. Paragraphs marks, partial words, and punctuation are included in the count.

```
MsgBox "There are " & Selection.Words.Count & " words."
```

This example steps through the words in myRange (which spans from the beginning of the active document to the end of the selection) and deletes the word "Franklin" (including the trailing space) wherever it occurs in the range.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=Selection.End)
For Each aWord In myRange.Words
    If aWord.Text = "Franklin " Then aWord.Delete
Next aWord
```



↳ [Show All](#)

WordWrap Property

▶ [WordWrap property as it applies to the Cell object.](#)

True if Microsoft Word wraps text to multiple lines and lengthens the cell so that the cell width remains the same. Read/write **Boolean**.

expression.**WordWrap**

expression Required. An expression that returns a [Cell](#) object.

▶ [WordWrap property as it applies to the Paragraph, ParagraphFormat, Paragraphs, and TextFrame objects.](#)

True if Microsoft Word wraps Latin text in the middle of a word in the specified paragraphs or text frames. This property returns **wdUndefined** if it's set to **True** for only some of the specified paragraphs or text frames. Read/write **Long**. This usage may not be available to you, depending on the language support (U.S. English, for example) that you've selected or installed.

expression.**WordWrap**

expression Required. An expression that returns one of the above objects.

Example

▶ [As it applies to the **Cell** object.](#)

This example sets Microsoft Word to wrap text to multiple lines in the third cell of the first table so that the cell's width remains the same.

```
ActiveDocument.Tables(1).Range.Cells(3).WordWrap = True
```

▶ [As it applies to the **Paragraph**, **ParagraphFormat**, **Paragraphs**, and **TextFrame** objects.](#)

This example sets Microsoft Word to wrap Latin text in the middle of a word in the first paragraph of the active document.

```
ActiveDocument.Paragraphs(1).WordWrap = True
```



WPDocNavKeys Property

True to enable in Microsoft Word navigation keys for WordPerfect users.
Read/write **Boolean**.

expression.**WPDocNavKeys**

expression Required. An expression that returns an [Options](#) object.

Example

This example sets Word to use WordPerfect navigation keys.

```
Sub WPNavKeys()  
    Options.WPDocNavKeys = True  
End Sub
```

This example returns the status of the **Navigation keys for WordPerfect users** option on the **General** tab in the **Options** dialog box (**Tools** menu).

```
Sub WPKeyStatus()  
    Dim bKeyStatus As Boolean  
  
    bKeyStatus = Options.WPDocNavKeys  
    MsgBox bKeyStatus  
End Sub
```



WPHelp Property

-
True if pressing Microsoft Word key combinations that produce actions in WordPerfect displays dialog boxes that describe how to perform the equivalent actions in Word. Read/write **Boolean**.

expression.**WPHelp**

expression Required. An expression that returns an [Options](#) object.

Example

This example toggles WordPerfect help between True and False.

```
Sub WPHelpToggle()  
    Options.WPHe1p = Not Options.WPHe1p  
End Sub
```

This example displays the status of the **Help for WordPerfect users** option on the **General** tab in the **Options** dialog box (**Tools** menu).

```
Sub WPHelpStatus()  
    MsgBox Options.WPHe1p  
End Sub
```



↳ [Show All](#)

Wrap Property

Returns or sets what happens if the search begins at a point other than the beginning of the document and the end of the document is reached (or vice versa if **Forward** is set to **False**) or if the search text isn't found in the specified selection or range. Read/write [WdFindWrap](#).

WdFindWrap can be one of these WdFindWrap constants.

wdFindAsk After searching the selection or range, Word displays a message asking whether to search the remainder of the document.

wdFindContinue The find operation continues when the beginning or end of the search range is reached.

wdFindStop The find operation ends when the beginning or end of the search range is reached.

expression.**Wrap**

expression Required. An expression that returns a [Find](#) object.

Example

The following example searches forward through the document for the word "aspirin." When the end of the document is reached, the search continues at the beginning of the document. If the word "aspirin" is found, it's selected.

```
Sub WordFind()  
  With Selection.Find  
    .Forward = True  
    .ClearFormatting  
    .MatchWholeWord = True  
    .MatchCase = False  
    .Wrap = wdFindContinue  
    .Execute FindText:="aspirin"  
  End With  
End Sub
```



WrapAroundText Property

Returns or sets whether text should wrap around the specified rows. Returns **wdUndefined** if only some of the specified rows have wrapping enabled. Can be set to **True** or **False**. Read/write **Long**.

expression.**WrapAroundText**

expression Required. An expression that returns a [Rows](#) object.

Remarks

Setting the **WrapAroundText** property to **False** also sets the [AllowOverlap](#) property to **False**. Setting the **AllowOverlap** property to **True** also sets the **WrapAroundText** property to **True**.

Example

This example sets Microsoft Word to wrap text around the first table in the document.

```
ActiveDocument.Tables(1).Rows.WrapAroundText = True
```



WrapFormat Property

-

Returns a [WrapFormat](#) object that contains the properties for wrapping text around the specified shape or shape range. Read-only.

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.

```
Set myOval = _
    ActiveDocument.Shapes.AddShape(msoShapeOval, 36, 36, 90, 50)
With myOval.WrapFormat
    .Type = wdWrapSquare
    .Side = wdWrapBoth
    .DistanceTop = InchesToPoints(0.1)
    .DistanceBottom = InchesToPoints(0.1)
    .DistanceLeft = InchesToPoints(0.1)
    .DistanceRight = InchesToPoints(0.1)
End With
```



WrapToWindow Property

True if lines wrap at the right edge of the document window rather than at the right margin or the right column boundary. Read/write **Boolean**.

expression.**WrapToWindow**

expression Required. An expression that returns a [View](#) object.

Remarks

This property has no effect in print layout or Web layout view.

Example

This example wraps the text to fit within the active window.

```
With ActiveDocument.ActiveWindow.View  
    .Type = wdNormalView  
    .WrapToWindow = True  
End With
```



WritePassword Property

-
Sets a password for saving changes to the specified document. Write-only **String**.

Example

If the active document isn't already protected against saving changes, this example sets "secret" as the write password for the document.

```
Set myDoc = ActiveDocument  
If myDoc.WriteReserved = False Then myDoc.WritePassword = "secret"
```



WriteReserved Property

-
True if the specified document is protected with a write password. Read-only **Boolean**.

Example

This example displays a message if the active document has a write password.

```
If ActiveDocument.WriteReserved = True Then  
    MsgBox "Changes cannot be made to this document."  
End If
```



WritingStyleList Property

Returns a string array that contains the names of all writing styles available for the specified language. Read-only **Variant**.

expression.**WritingStyleList**

expression Required. An expression that returns a [Language](#) object.

Example

This example displays each writing style available for U.S. English. Each writing style and its number in the array are also displayed in the Immediate window of the Visual Basic editor.

```
Sub WritingStyles()  
    Dim WrStyles As Variant  
    Dim i As Integer  
  
    WrStyles = Languages(wdEnglishUS).WritingStyleList  
    For i = 1 To UBound(WrStyles)  
        MsgBox WrStyles(i)  
        Debug.Print WrStyles(i) & " [" & Trim(Str$(i)) & "]"  
    Next i  
End Sub
```



Yellow Property

-
Sets or returns a **Long** that represents the yellow component of a CMYK color.
Read-only.

expression.**Yellow**

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

Example

This example creates a new shape, then retrieves the four CMYK values from an existing shape in the active document, and then sets the CMYK fill color of the new 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.Shapes(1)  
    Set shpStar = ActiveDocument.Shapes.AddShape _  
        (Type:=msoShape5pointStar, Left:=200, _  
         Top:=100, Width:=150, Height:=150)  
  
    'Get current shapes CMYK colors  
    With shpHeart.Fill.ForeColor  
        lngCyan = .Cyan  
        lngMagenta = .Magenta  
        lngYellow = .Yellow  
        lngBlack = .Black  
    End With  
  
    'Set new shape to current shapes CMYK colors  
    shpStar.Fill.ForeColor.SetCMYK _  
        Cyan:=lngCyan, Magenta:=lngMagenta, _  
        Yellow:=lngYellow, Black:=lngBlack  
End Sub
```



Zoom Property

Returns a [Zoom](#) object that represents the magnification for the specified view.

expression.**Zoom**

expression Required. An expression that returns one of a [View](#) object.

Example

This example changes the zoom percentage of each open window to 125 percent.

```
Sub wndBig()  
    Dim wndBig As Window  
  
    For Each wndBig In Windows  
        wndBig.View.Zoom.Percentage = 125  
    Next wndBig  
End Sub
```

This example changes the zoom percentage of the active window so that the entire width of the text is visible.

```
ActiveDocument.ActiveWindow.View.Zoom.PageFit = wdPageFitBestFit
```



Zooms Property

Returns a [Zooms](#) collection that represents the magnification options for each view (normal view, outline view, print layout view, and so on).

expression.**Zooms**

expression Required. An expression that returns a [Pane](#) object.

Remarks

For information about returning a single member of a collection, see [Returning an Object from a Collection](#).

Example

This example sets the magnification in normal view to 100 percent for each open window.

```
Dim wndLoop as Window
```

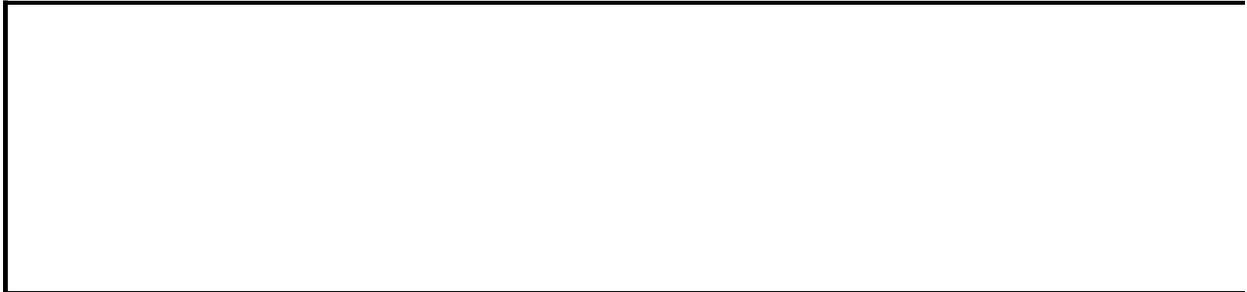
```
For Each wndLoop In Windows
```

```
    wndLoop.ActivePane.Zooms(wdNormalView).Percentage = 100
```

```
Next wndLoop
```

This example sets the magnification in print layout view so that an entire page is visible.

```
ActiveDocument.ActiveWindow.Panes(1).Zooms(wdPrintView).PageFit = _  
    wdPageFitFullPage
```



ZOrderPosition Property

Returns the position of the specified shape in the z-order. `Shapes(1)` returns the shape at the back of the z-order, and `Shapes(Shapes.Count)` returns the shape at the front of the z-order. Read-only **Long**.

This property is read-only. To set the shape's position in the z-order, use the [ZOrder](#) method.

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 `myDocument`, the expression `myDocument.Shapes(1)` returns the shape at the back of the z-order, and the expression `myDocument.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.

Example

This example adds an oval to myDocument and then places the oval second from the back in the z-order if there is at least one other shape on the document.

```
Set myDocument = ActiveDocument
With myDocument.Shapes.AddShape(msoShapeOval, 100, 100, 100, 300)
    While .ZOrderPosition > 2
        .ZOrder msoSendBackward
    Wend
End With
```



Close Event

-

Occurs when a document is closed.

Private Sub Document_Close()

Remarks

If the event procedure is stored in a template, the procedure will run when a new document based on that template is closed and when the template itself is closed (after being opened as a document).

For information about using events with a **Document** object, see [Using Events with the Document Object](#).

Example

This example makes a backup copy of the document on a file server when the document is closed. (The procedure can be stored in the **ThisDocument** class module of a document or its attached template.)

```
Private Sub Document_**Close()  
    ThisDocument.Save  
    ThisDocument.SaveAs "\\network\backup\" & ThisDocument.Name  
End Sub
```



DocumentBeforeClose Event

-
Occurs immediately before any open document closes.

Private Sub *object*_DocumentBeforeClose(**ByVal** *Doc* **As Document**, *Cancel* **As Boolean**)

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 document that's being closed.

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.

Example

This example prompts the user for a yes or no response before closing any document. 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](#) Using Events with the Application Object for directions on how to accomplish this.

```
Public WithEvents appWord as Word.Application

Private Sub appWord_DocumentBeforeClose _
    (ByVal Doc As Document, _
    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
```



DocumentBeforePrint Event

-
Occurs before any open document is printed.

Private Sub *object*_DocumentBeforePrint(**ByVal** *Doc* **As** Document, *Cancel* **As** Boolean)

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 document that's being printed.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the document isn't printed when the procedure is finished.

Example

This example prompts the user for a yes or no response before printing any document. 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 appWord as Word.Application
```

```
Private Sub appWord_DocumentBeforePrint _  
    (ByVal Doc As Document, _  
    Cancel As Boolean)
```

```
    Dim intResponse As Integer
```

```
    intResponse = MsgBox("Have you checked the " _  
        & "printer for letterhead?", _  
        vbYesNo)
```

```
    If intResponse = vbNo Then Cancel = True  
End Sub
```



DocumentBeforeSave Event

-
Occurs before any open document is saved.

Private Sub *object*_DocumentBeforeSave(**ByVal** *Doc* **As** Document, **ByVal** *SaveAsUI* **As** Boolean, *Cancel* **As** Boolean)

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 document that's being saved.

SaveAsUI **True** to display the **Save As** dialog box.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the document isn't saved when the procedure is finished.

Example

This example prompts the user for a yes or no response before saving any document. 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 appWord as Word.Application
```

```
Private Sub appWord_DocumentBeforeSave _  
    (ByVal Doc As Document, _  
    ByVal SaveAsUI As Boolean, _  
    Cancel As Boolean)
```

```
    Dim intResponse As Integer
```

```
    intResponse = MsgBox("Do you really want to " _  
        & "save the document?", _  
        vbYesNo)
```

```
    If intResponse = vbNo Then Cancel = True
```

```
End Sub
```



DocumentChange Event

-
Occurs when a new document is created, when an existing document is opened, or when another document is made the active document.

Private Sub *object*_DocumentChange()

object An object of type **Application** declared with events in a class module. For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Example

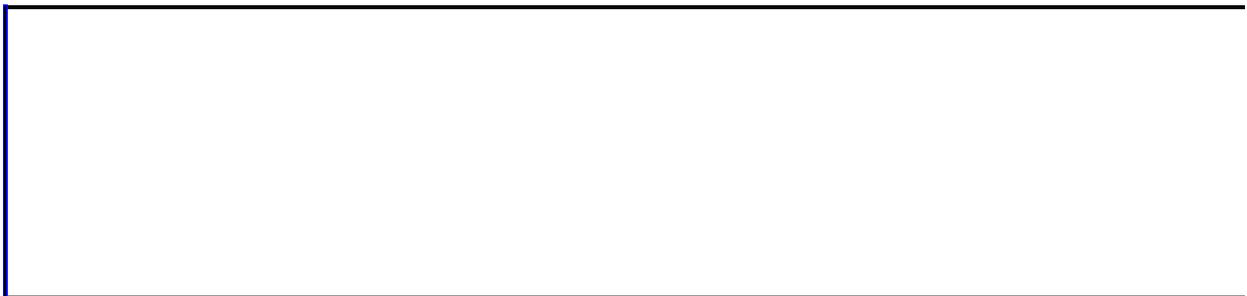
This example asks the user whether to save all the other open documents when the document focus changes. This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see [Using Events with the Application Object](#) for directions on how to accomplish this.

```
Public WithEvents appWord as Word.Application

Private Sub appWord_DocumentChange()
    Dim intResponse As Integer
    Dim strName As String
    Dim docLoop As Document

    intResponse = MsgBox("Save all other documents?", vbYesNo)

    If intResponse = vbYes Then
        strName = ActiveDocument.Name
        For Each docLoop In Documents
            With docLoop
                If .Name <> strName Then
                    .Save
                End If
            End With
        Next docLoop
    End If
End Sub
```



DocumentOpen Event

-
Occurs when a document is opened.

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 **Application** object, see [Using Events with the Application Object](#).

Doc The document that's being opened.

Example

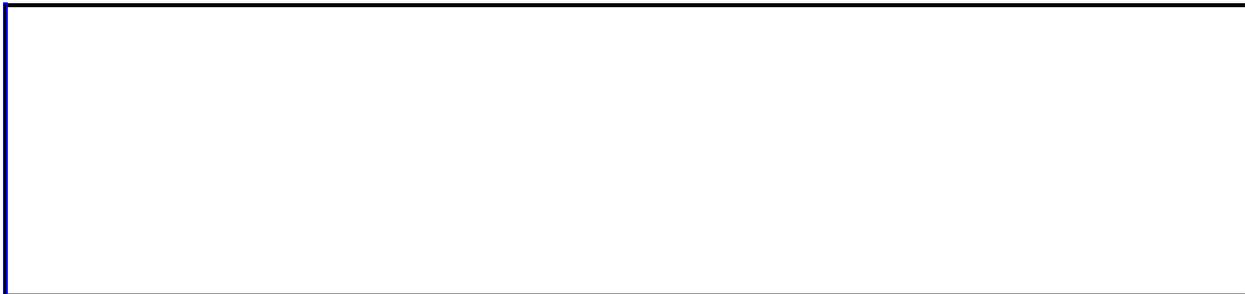
This example asks the user whether to save all other open documents when a document is opened. This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see [Using Events with the Application Object](#) for directions on how to accomplish this.

```
Public WithEvents appWord as Word.Application

Private Sub appWord_DocumentOpen(ByVal Doc As Document)
    Dim intResponse As Integer
    Dim strName As String
    Dim docLoop As Document

    intResponse = MsgBox("Save all other documents?", vbYesNo)

    If intResponse = vbYes Then
        strName = ActiveDocument.Name
        For Each docLoop In Documents
            With docLoop
                If .Name <> strName Then
                    .Save
                End If
            End With
        Next docLoop
    End If
End Sub
```



↳ [Show All](#)

EPostageInsert Event

-
Occurs when a user inserts electronic postage into a document.

Private Sub *object*_EPostageInsert(ByVal *Doc* As Document)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The name of the document to which to add electronic postage.

Example

This example displays a message when electronic postage is inserted into a document.

```
Private Sub AppWord_EPostageInsert(ByVal Doc As Document)
    MsgBox "You just inserted electronic postage into your document."
End Sub
```



↳ [Show All](#)

EPostagePropertyDialog Event

-

Occurs when a user clicks the **E-postage Properties (Labels and Envelopes** dialog box) button or **Print Electronic Postage** toolbar button. This event allows a third-party software application to intercept and show their properties dialog box.

Private Sub *object*_EPostagePropertyDialog(**ByVal** *Doc* As Document)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The name of the document to which to add electronic postage.

Example

This example displays a message when a user clicks on either the **Add Electronic Postage** or **Print Electronic Postage** button.

```
Private Sub AppWord_EPostagePropertyDialog(ByVal Doc As Document)
    MsgBox "You have clicked on a button to " & _
        "display the ePostage property dialog box."
End Sub
```



↳ [Show All](#)

MailMergeAfterMerge Event

-
Occurs after all records in a mail merge have merged successfully.

Private Sub *object*_MailMergeAfterMerge(ByVal *Doc* As Document, ByVal *DocResult* As Document)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

DocResult The document created from the mail merge.

Example

This example displays a message stating that all records in the specified document are finished merging. If the document has been merged to a second document, the message includes the name of the new document. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

```
Private Sub MailMergeApp_MailMergeAfterMerge(ByVal Doc As Document,
    ByVal DocResult As Document)
    If DocResult Is Nothing Then
        MsgBox "Your mail merge on " & _
            Doc.Name & " is now finished."

    Else
        MsgBox "Your mail merge on " & _
            Doc.Name & " is now finished and " & _
            DocResult.Name & " has been created."
    End If
End Sub
```



↳ [Show All](#)

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 An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

Example

This example displays a message with the value of the first and second fields in the record that has just finished merging. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

```
Private Sub MailMergeApp_MailMergeAfterRecordMerge(ByVal Doc As Docu

    With Doc.MailMerge.DataSource
        MsgBox .DataFields(1).Value & " " & _
            .DataFields(2).Value & " is finished merging."
    End With

End Sub
```



↳ [Show All](#)

MailMergeBeforeMerge Event

Occurs when a merge is executed before any records merge.

Private Sub *object*_MailMergeBeforeMerge(ByVal *Doc* As Document, ByVal *StartRecord* As Long, ByVal *EndRecord* As Long, *Cancel* As Boolean)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

StartRecord The first record in the data source to include in the mail merge.

EndRecord The last record in the data source to include in the mail merge.

Cancel **True** stops the mail merge process before it starts.

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. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

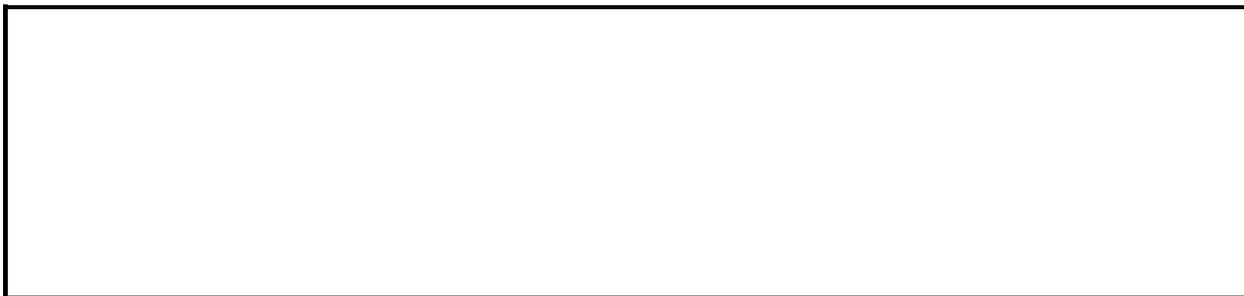
```
Private Sub MailMergeApp_MailMergeBeforeMerge(ByVal Doc As Document,
    ByVal StartRecord As Long, ByVal EndRecord As Long, _
    Cancel As Boolean)

    Dim intVBAnswer As Integer

    '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, "MailMergeBeforeMerge E

    'If users response to question is No, cancel the merge process
    'and deliver a message to the user stating the merge is cancelled
    If intVBAnswer = vbNo Then
        Cancel = True
        MsgBox "You have cancelled mail merge for " & _
            Doc.Name & "."
    End If

End Sub
```



↳ [Show All](#)

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 An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

Cancel **True** stops the mail merge process for the current record only before it starts.

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. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

```
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
```



↳ [Show All](#)

MailMergeDataSourceLoad Event

-
Occurs when the data source is loaded for a mail merge.

Private Sub *object*_MailMergeDataSourceLoad(**ByVal** *Doc* As Document)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

Example

This example displays a message with the data source file name when the data source starts loading. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

```
Private Sub MailMergeApp_MailMergeDataSourceLoad(ByVal Doc As Docume
    Dim strDSName As String
    Dim intDSLenght As Integer
    Dim intDSStart As Integer

    'Extract from the Name property only the filename
    intDSLenght = Len(Doc.MailMerge.DataSource.Name)
    intDSStart = InStrRev(Doc.MailMerge.DataSource.Name, "\")
    intDSStart = intDSLenght - intDSStart
    strDSName = Right(Doc.MailMerge.DataSource.Name, intDSStart)

    'Deliver a message to user when data source is loading
    MsgBox "Your data source, " & strDSName & ", is now loading."
End Sub
```



↳ [Show All](#)

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 An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

Handled **True** runs the accompanying validation code against the mail merge data source. **False** cancels the data source validation.

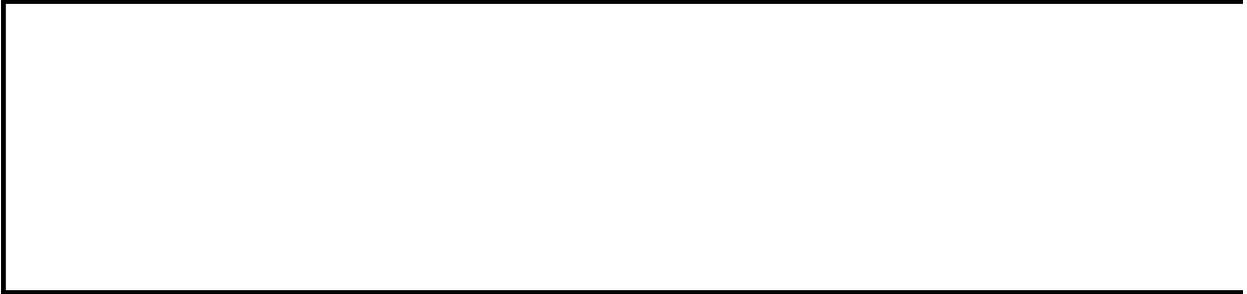
Remarks

If you don't have address verification software installed on your computer, the **MailMergeDataSourceValidate** event allows you to create simple filtering routines, such as looping through records to check the postal codes and removing 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 Microsoft Visual Basic commands to search for text or special characters.


```
        'Move the record to the next record in the data source
        .ActiveRecord = wdNextRecord

    'End the loop when the counter variable
    'equals the number of records in the data source
    Loop Until intCount = .RecordCount
End With
```

End Sub



↳ [Show All](#)

MailMergeWizardSendToCustom Event

-
Occurs when the custom button is clicked on step six of the Mail Merge Wizard.

Private Sub *object*_MailMergeWizardSendToCustom(ByVal *Doc* As Document)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

Remarks

Use the [ShowSendToCustom](#) property to create a custom button on the sixth step of the Mail Merge Wizard.

Example

This example executes a merge to a fax machine when a user clicks the custom destination button. This example assumes that the user has access to a custom destination button, fax numbers are included for each record in the data source, and an application variable called MailMergeApp has been declared and set equal to the Word **Application** object.

```
Private Sub MailMergeApp_MailMergeWizardSendToCustom(ByVal Doc As Do  
    With Doc.MailMerge  
        .Destination = wdSendToFax  
        .Execute  
    End With  
End Sub
```



↳ [Show All](#)

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, *ToState* As Long, *Handled* As Boolean)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

Doc The mail merge main document.

FromState The Mail Merge Wizard step from which a user is moving.

ToState The Mail Merge Wizard step to which a user is moving.

Handled True moves the user to the next step. **False** for the user to remain at the current step.

Example

This example displays a message when a user moves from step three of the Mail Merge Wizard to step four. Based on the answer to the message, the user will either move to step four or remain at step three. This example assumes that you have declared an application variable called MailMergeApp in your general declarations and have set the variable equal to the Word **Application** object.

```
Private Sub MailMergeApp_MailMergeWizardStateChange(ByVal Doc As Doc
    FromState As Long, ToState As Long, Handled As Boolean)

    Dim intVBAnswer As Integer
    FromState = 3
    ToState = 4

    'Display a message when moving from step three to step four
    intVBAnswer = MsgBox("Have you selected all of your recipients?"
        vbYesNo, "Wizard State Event!")

    If intVBAnswer = vbYes Then
        'Continue on to step four
        Handled = True
    Else
        'Return to step three
        MsgBox "Please select all recipients to whom " & _
            "you want to send this letter."
        Handled = False
    End If

End Sub
```



New Event

-

Occurs when a new document based on the template is created. A procedure for the **New** event will run only if it is stored in a template.

Private Sub Document_New()

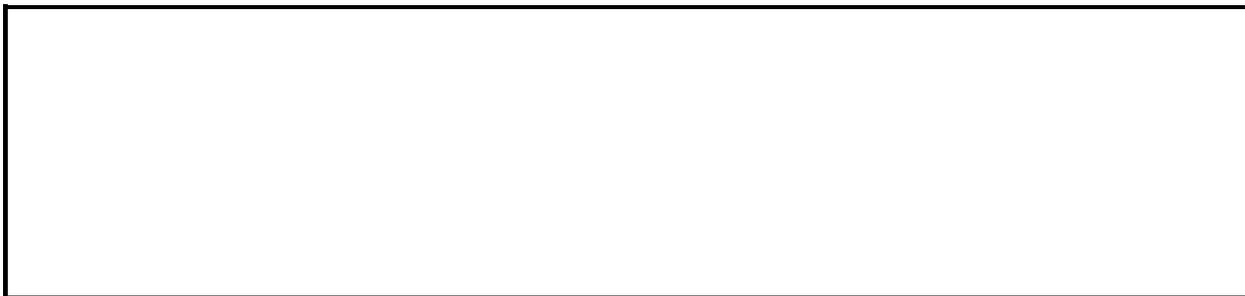
Remarks

For information about using events with the **Document** object, see [Using Events with the Document Object](#).

Example

This example asks the user whether to save all other open documents when a new document based on the template is created. (This procedure is stored in the **ThisDocument** class module of a template, not a document.)

```
Private Sub Document_New()  
    Dim intResponse As Integer  
    Dim strName As String  
    Dim docLoop As Document  
  
    intResponse = MsgBox("Save all other documents?", vbYesNo)  
  
    If intResponse = vbYes Then  
        strName = ActiveDocument.Name  
        For Each docLoop In Application.Documents  
            With docLoop  
                If .Name <> strName Then  
                    .Save  
                End If  
            End With  
        Next docLoop  
    End If  
End Sub
```



NewDocument Event

-
Occurs when a new document 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

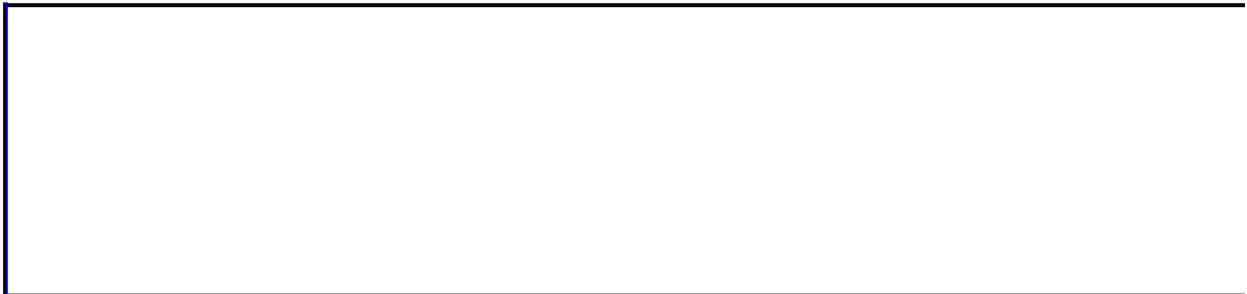
This example asks the user whether to save all other open documents when a new document is created. This code must be placed in a class module, and an instance of the class must be correctly initialized in order to see this example work; see [Using Events with the Application Object](#) for directions on how to accomplish this.

```
Public WithEvents appWord as Word.Application

Private Sub appWord_NewDocument(ByVal Doc As Document)
    Dim intResponse As Integer
    Dim strName As String
    Dim docLoop As Document

    intResponse = MsgBox("Save all other documents?", vbYesNo)

    If intResponse = vbYes Then
        strName = ActiveDocument.Name
        For Each docLoop In Documents
            With docLoop
                If .Name <> strName Then
                    .Save
                End If
            End With
        Next docLoop
    End If
End Sub
```



Open Event

-

Occurs when a document is opened.

Private Sub Document_Open()

Remarks

If the event procedure is stored in a template, the procedure will run when a new document based on that template is opened and when the template itself is opened as a document.

For information about using events with the **Document** object, see [Using Events with the Document Object](#).

Example

This example displays a message when a document is opened. (The procedure can be stored in the **ThisDocument** class module of a document or its attached template.)

```
Private Sub Document_Open()  
    MsgBox "This document is copyrighted."  
End Sub
```



Quit Event

-
Occurs when the user quits Word.

Private Sub *object*_Quit()

object An object of type **Application** declared with events in a class module. 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 Word. As a result, when Word 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 appWord as Word.Application

Private Sub appWord_Quit()
    CommandBars("Standard").Visible = True
    CommandBars("Formatting").Visible = True
End Sub
```



WindowActivate Event

-
Occurs when any document window is activated.

Private Sub *object*_WindowActivate(ByVal *Doc* As Word.Document, ByVal *Wn* As Word.Window)

object An object of type **Application** declared with events in a class module. For more information about using events with the **Application** object or the **Document** object, see [Using Events with the Application Object](#) or [Using Events with the Document Object](#).

Doc Used only with the **Application** object. The document displayed in the activated window.

Wn The window that's being activated.

Example

This example maximizes any document 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 appWord as Word.Application
```

```
Private Sub appWord_WindowActivate _  
    (ByVal Wn As Word.Window)  
    Wn.WindowState = wdWindowStateMaximize  
End Sub
```



WindowBeforeDoubleClick Event

-

Occurs when the editing area of a document window is double-clicked, before the default double-click action.

Private Sub *object*_WindowBeforeDoubleClick(**ByVal** *Sel* As Selection, **ByVal** *Cancel* As Boolean)

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](#).

Sel The current selection.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default double-click action does not occur when the procedure is finished.

Example

This example prompts the user for a yes or no response before executing the default double-click action. 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 appWord as Word.Application

Private Sub appWord_WindowBeforeDoubleClick _
    (ByVal Sel As Selection, _
    Cancel As Boolean)
    Dim intResponse As Integer
    intResponse = MsgBox("Selection = " & Sel & vbCrLf & vbCrLf _
        & "Continue with operation on this selection?", _
        vbYesNo)
    If intResponse = vbNo Then Cancel = True
End Sub
```



WindowBeforeRightClick Event

-

Occurs when the editing area of a document window is right-clicked, before the default right-click action.

Private Sub *object*_WindowBeforeRightClick(ByVal *Sel* As Selection, ByVal *Cancel* As Boolean)

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](#).

Sel The current selection.

Cancel **False** when the event occurs. If the event procedure sets this argument to **True**, the default right-click action does not occur when the procedure is finished.

Example

This example prompts the user for a yes or no response before executing the default right-click action. 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 appWord as Word.Application

Private Sub appWord_WindowBeforeRightClick _
    (ByVal Sel As Selection, _
    Cancel As Boolean)
    Dim intResponse As Integer

    intResponse = MsgBox("Selection = " & Sel & vbCrLf & vbCrLf _
        & "Continue with operation on this selection?", _
        vbYesNo)
    If intResponse = vbNo Then Cancel = True
End Sub
```



WindowDeactivate Event

-
Occurs when any document window is deactivated.

Private Sub *object*_WindowDeactivate(ByVal *Doc* As Word.Document, ByVal *Wn* As Word.Window)

object An object of type **Application** declared with events in a class module. For more information about using events with the **Application** object or the **Document** object, see [Using Events with the Application Object](#) or [Using Events with the Document Object](#).

Doc Used only with the **Application** object. The document displayed in the deactivated window.

Wn The deactivated window.

Example

This example minimizes any document 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 appWord as Word.Application
```

```
Private Sub appWord_WindowDeactivate _  
    (ByVal Wn As Word.Window)  
    Wn.WindowState = wdWindowStateMinimize  
End Sub
```



WindowSelectionChange Event

-
Occurs when the selection changes in the active document window.

Private Sub *object*_WindowSelectionChange(ByVal *Sel* As Selection)

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](#).

Sel The new selection.

Example

This example applies bold formatting to the new selection. 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 appWord as Word.Application
```

```
Private Sub appWord_WindowSelectionChange _  
    (ByVal Sel As Selection)  
    Sel.Font.Bold = True  
End Sub
```



↳ [Show All](#)

WindowSize Event

-
Occurs when the application window is resized or moved.

Private Sub *object*_WindowSize(**ByVal** *Doc* As Document, **ByVal** *Wn* As Window)

object An object of type **Application** declared with events in a [class module](#). For information about using events with the **Application** object, see [Using Events with the Application Object](#).

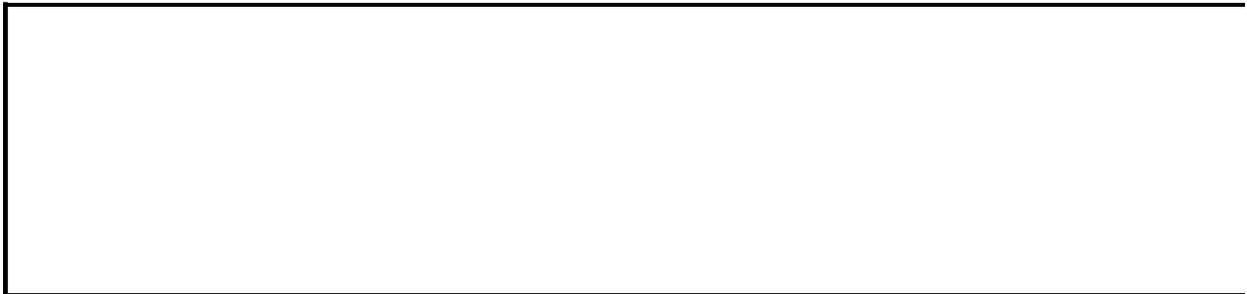
Doc The document in the window being sized.

Wn The window being sized.

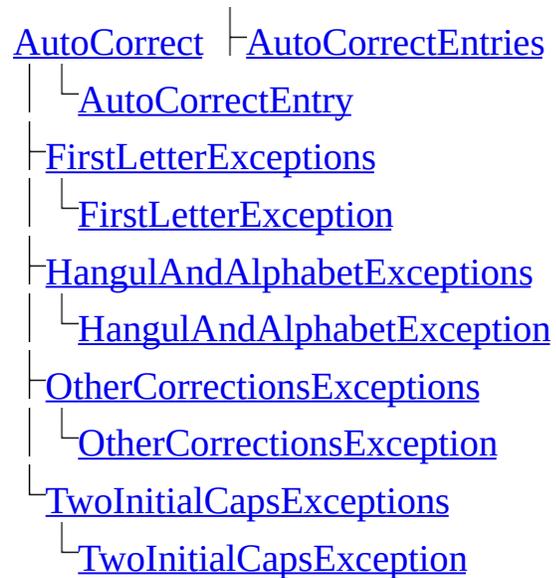
Example

This example displays a message every time the Microsoft Word application window is moved or resized. This example assumes that you have declared an application variable called "WordApp" in your general declarations and have set the variable equal to the Word Application object.

```
Private Sub WordApp_WindowSize(ByVal Doc As Document, _  
    ByVal Wn As Window)  
    MsgBox "You have just resized or moved your window."  
End Sub
```



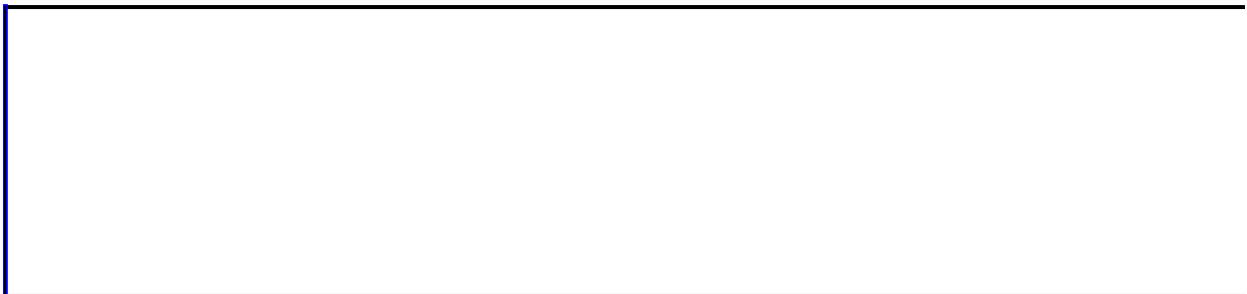
Microsoft Word Objects (AutoCorrect)



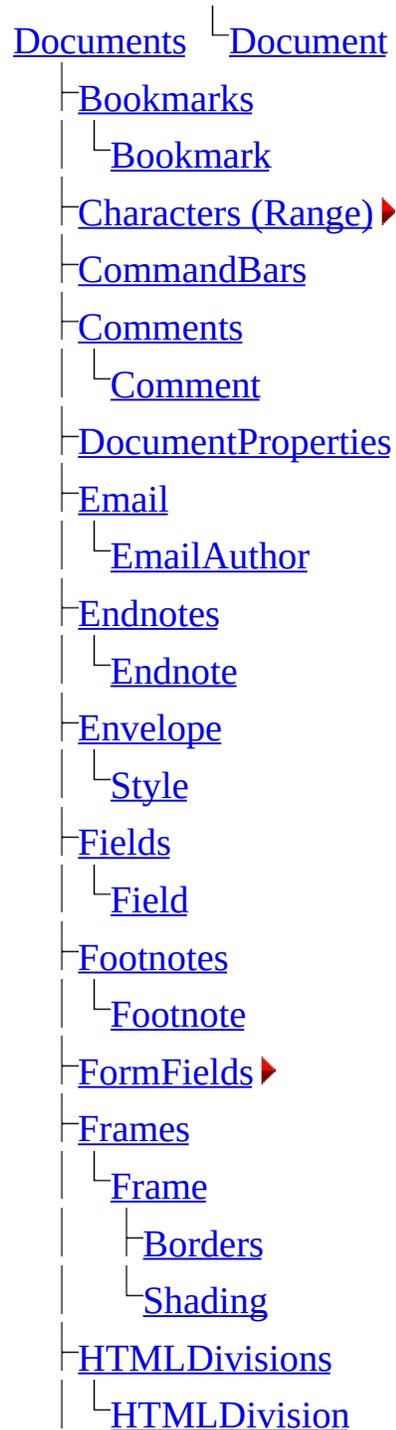
Legend

Object and collection

Object only



Microsoft Word Objects (Documents)



- └─ [Borders](#)
- ─ [HTMLProject](#)
- ─ [Hyperlinks](#)
 - └─ [Hyperlink](#)
- ─ [Indexes](#)
 - └─ [Field](#)
 - └─ [Index](#)
- ─ [InlineShapes](#) ▶
- ─ [LetterContent](#)
- ─ [ListParagraphs](#)
 - └─ [Paragraph](#) ▶
- ─ [Lists](#)
 - └─ [List](#)
 - └─ [ListParagraphs](#)
 - └─ [Range](#)
 - └─ [ListParagraphs](#)
 - └─ [Range](#)
- ─ [ListTemplates](#) ▶
- ─ [MailMerge](#) ▶
- ─ [PageSetup](#)
 - └─ [LineNumbering](#)
 - └─ [TextColumns](#)
 - └─ [TextColumn](#)
- ─ [Paragraphs](#) ▶
- ─ [ProofreadingErrors \(Range\)](#) ▶
- ─ [Range](#) ▶
- ─ [ReadabilityStatistics](#)
 - └─ [ReadabilityStatistic](#)
- ─ [Revisions](#)
 - └─ [Revision](#)

- └ [RoutingSlip](#)
- └ [Scripts](#)
 - └ [Script](#)
- └ [Sections](#) ▶
- └ [Sentences \(Range\)](#) ▶
- └ [Shapes](#) ▶
- └ [SignatureSet](#)
 - └ [Signature](#)
- └ [SmartTags](#)
 - └ [SmartTag](#)
- └ [StoryRanges \(Range\)](#) ▶
- └ [Styles](#) ▶
- └ [StyleSheets](#)
 - └ [StyleSheet](#)
- └ [Subdocuments](#)
 - └ [Subdocument](#)
- └ [Tables](#)
 - └ [Table](#)
- └ [TablesOfAuthorities](#)
 - └ [Field](#)
 - └ [TableOfAuthorities](#)
- └ [TablesOfAuthoritiesCategories](#)
 - └ [TableOfAuthoritiesCategory](#)
- └ [TablesOfContents](#)
 - └ [Field](#)
 - └ [TableOfContents](#)
 - └ [HeadingStyles](#)
- └ [TablesOfFigures](#)
 - └ [Field](#)
 - └ [TableOfFigures](#)

- | [Variables](#)
- | | [Variable](#)
- | [Versions](#)
- | | [Version](#)
- | [WebOptions](#)
- | [Windows](#) ▶
- | [Words \(Range\)](#) ▶

Legend

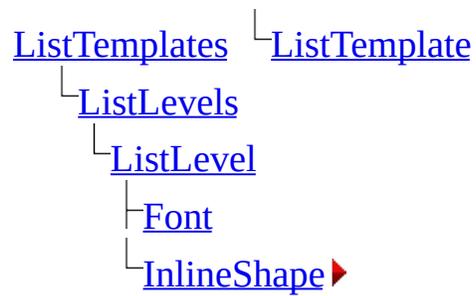
Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (ListTemplates)

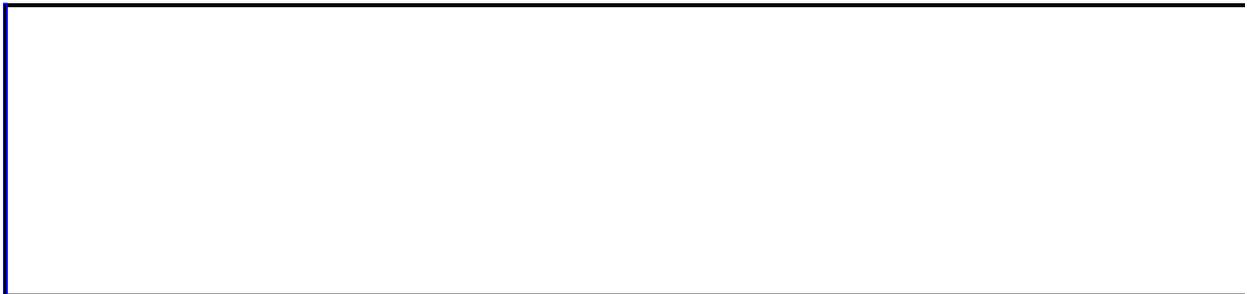


Legend

Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (Selection)

[Selection](#) | [AutoTextEntry](#)

| [Bookmarks](#)

| | [Bookmark](#)

| [Borders](#)

| | [Border](#)

| [Cells](#) ▶

| [Characters \(Range\)](#) ▶

| [Columns](#)

| | [Column](#)

| [Comments](#)

| | [Comment](#)

| [Document](#) ▶

| [EndnoteOptions](#)

| [Endnotes](#)

| | [Endnote](#)

| [Fields](#)

| | [Field](#)

| [Find](#) ▶

| [Font](#)

| [FootnoteOptions](#)

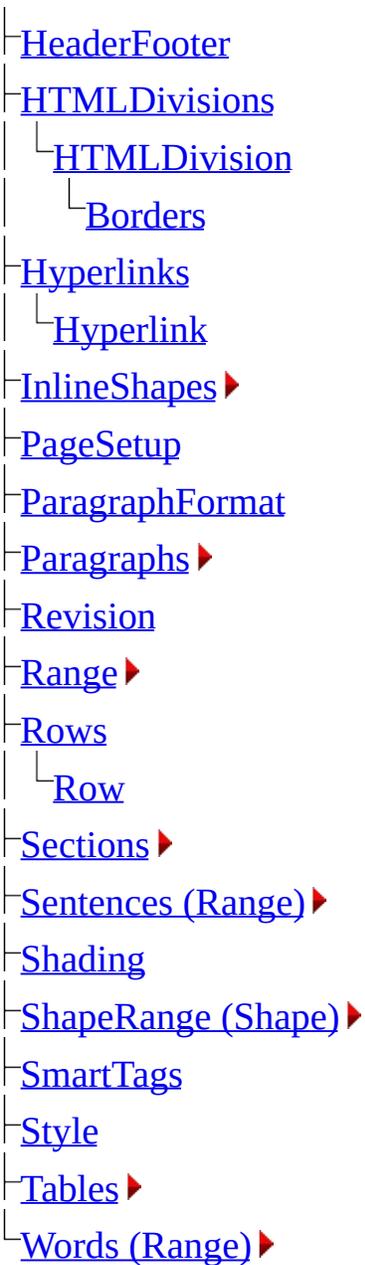
| [Footnotes](#)

| | [Footnote](#)

| [FormFields](#) ▶

| [Frames](#)

| | [Frame](#)



Legend

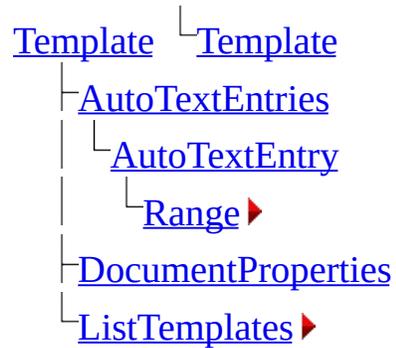
Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (Templates)



Legend

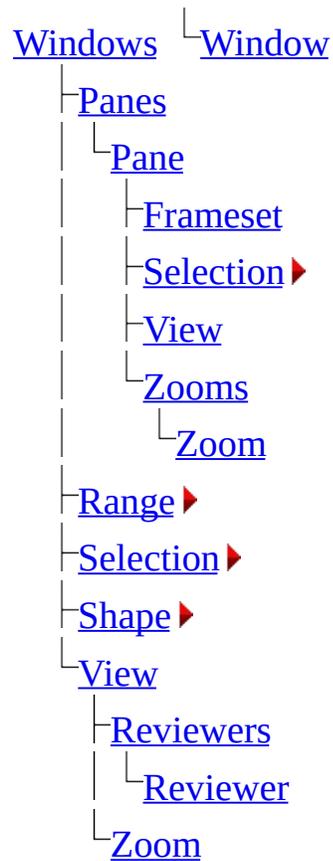
Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (Windows)



Legend

Object and collection

Object only

▶ Click red arrow to expand chart



↳ [Show All](#)

New Objects

Visit the [Office Developer Center](#) at MSDN Online for the latest Microsoft Word development information, including new technical articles, downloads, samples, product news, and more.

Objects that were added to Visual Basic in Microsoft Word 2002 are listed in the following table.

Object	Description
<u>CanvasShapes</u>	A collection of the shapes in a drawing canvas.
<u>ConditionalStyle</u>	Consists of special formatting applied to specified areas of a table when the selected table is formatted with a specified table style.
<u>CustomProperties</u>	A collection of custom properties related to a smart tag.
<u>CustomProperty</u>	A single custom property for a smart tag.
<u>Diagram</u>	A single diagram in a document.
<u>DiagramNode</u>	A single diagram node within a diagram.
<u>DiagramNodeChildren</u>	A collection of DiagramNode objects that represents the child nodes in a diagram.
<u>DiagramNodes</u>	A collection of DiagramNode objects that represent all the nodes in a diagram.
<u>EmailSignatureEntries</u>	A collection of all the e-mail signature entries available in Word.
<u>EmailSignatureEntry</u>	A single e-mail signature entry.
<u>EndnoteOptions</u>	Properties assigned to a range or selection of endnotes in a document.
<u>FootnoteOptions</u>	Properties assigned to a range or selection of footnotes in a document.
<u>HTMLDivision</u>	A single HTML division that can be added to a Web document.

HTMLDivisions

A collection of the HTML divisions that exist in a Web document

MappedDataField

A single mapped data field.

MappedDataFields

A collection of all the [mapped data fields](#) available in Microsoft Word.

Reviewer

A single reviewer of a document in which changes have been tracked.

Reviewers

A collection of reviewers who have reviewed one or more documents.

SmartTag

A string in a document or range that contains recognized type information.

SmartTags

A collection of **SmartTag** objects that represent text in a document that is marked as containing recognized type information.

Stylesheet

A single cascading style sheet attached to a web document.

Stylesheets

A collection of the cascading style sheets attached to a document.

TableStyle

A single style that contains formatting that can be applied to a table.

TaskPane

A single task pane available in Microsoft Word that contains commonly performed tasks.

TaskPanes

A collection of task panes that contains commonly-performed tasks in Microsoft Word.

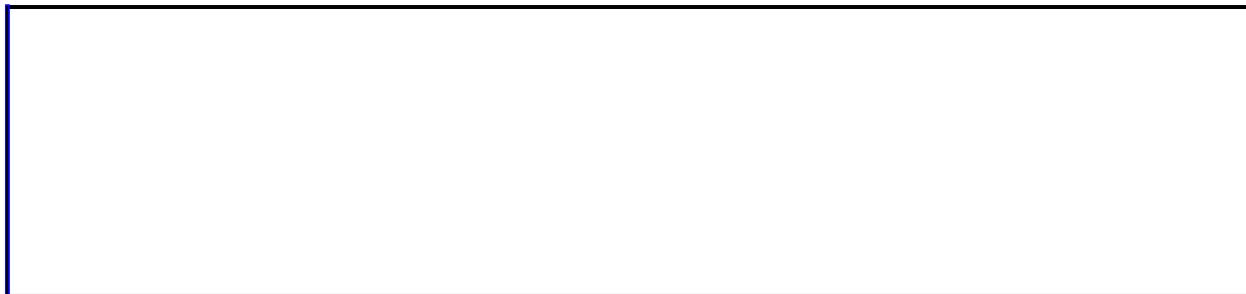
New Properties (by Object)

Properties that have been added to existing objects in Microsoft Word 2002 are listed in the following table (sorted by object name).

Object	New Properties
Application	AutoCorrectEmail , AutomationSecurity , DefaultLegalBlackline , EmailTemplate , FileDialog , NewDocument , ShowStartupDialog , ShowWindowsInTaskbar , TaskPanes
AutoCorrect	CorrectTableCells , DisplayAutoCorrectOptions
ColorFormat	TintAndShade , Black , Cyan , Ink , Magenta , OverPrint , Yellow
DefaultWebOptions	SaveNewWebPagesAsWebArchives , TargetBrowser
Diagram	AutoFormat , AutoLayout , Reverse
DiagramNode	Children , Diagram , Layout , Root , TextShape
DiagramNodeChildren	FirstChild , LastChild
Document	DefaultTableStyle , DefaultTargetFrame , DisableFeatures , DisableFeaturesIntroducedAfter , DoNotEmbedSystemFonts , EmbedLinguisticData , EmbedSmartTag , FormattingShowClear , FormattingShowFilter , FormattingShowFont , FormattingShowNumbering , FormattingShowParagraph , HTMLDivisions , MailEnvelope , PasswordEncryptionAlgorithm , PasswordEncryptionFileProperties , PasswordEncryptionKeyLength , PasswordEncryptionProvider , RemovePersonalInformation , Signatures , SmartTags , SmartTagsAsXMLProps , StyleSheets , TextEncoding , TextLineEnding
EmailOptions	EmbedSmartTags , HTMLFidelity , NewColorOnReply , PlainTextStyle ,

EmailSignature	<u>UseThemeStyleOnReply</u> <u>EmailSignatureEntries</u> <u>RecipientNameFromLeft</u> , <u>RecipientNameFromTop</u> , <u>RecipientPostalFromLeft</u> ,
Envelope	<u>RecipientPostalFromTop</u> , <u>SenderNameFromLeft</u> , <u>SenderNameFromTop</u> , <u>SenderPostalFromLeft</u> , <u>SenderPostalFromTop</u> , <u>Vertical</u>
Global	<u>AutoCorrectEmail</u>
InlineShape	<u>IsPictureBullet</u>
LineFormat	<u>InsetPen</u>
ListFormat	<u>ListPictureBullet</u>
ListLevel	<u>PictureBullet</u>
MailingLabel	<u>Vertical</u>
MailMerge	<u>HighlightMergeFields</u> , <u>MailFormat</u> , <u>ShowSendToCustom</u> , <u>WizardState</u>
MailMergeDataSource	<u>Included</u> , <u>InvalidAddress</u> , <u>InvalidComments</u> , <u>MappedDataFields</u> , <u>RecordCount</u> , <u>TableName</u>
MappedDataField	<u>DataFieldIndex</u> , <u>DataFieldName</u>
OLEFormat	<u>PreserveFormattingOnUpdate</u> <u>AutoCreateNewDrawings</u> , <u>BackgroundOpen</u> , <u>CommentColor</u> , <u>CtrlClickHyperlinkToOpen</u> , <u>DefaultEPostageApp</u> , <u>DefaultTextEncoding</u> , <u>DisableFeaturesByDefault</u> , <u>DisableFeaturesIntroducedAfterByDefault</u> , <u>DisplayPasteOptions</u> , <u>DisplaySmartTagButtons</u> , <u>FormatScanning</u> , <u>LabelSmartTags</u> , <u>LocalNetworkFile</u> , <u>PasteAdjustParagraphSpacing</u> , <u>PasteAdjustTableFormatting</u> , <u>PasteAdjustWordSpacing</u> , <u>PasteMergeFromPPT</u> , <u>PasteMergeFromXL</u> , <u>PasteMergeLists</u> , <u>PasteSmartCutPaste</u> , <u>PasteSmartStyleBehavior</u> , <u>PictureWrapType</u> , <u>PromptUpdateStyle</u> , <u>RevisionsBalloonPrintOrientation</u> , <u>SequenceCheck</u> , <u>ShowFormatError</u> , <u>SmartParaSelection</u> , <u>StoreRSIDOnSave</u> , <u>TypeNReplace</u> , <u>WarnBeforeSavingPrintingSendingMarkup</u>
Options	

PageSetup	BookFoldPrinting , BookFoldPrintingSheets , BookFoldRevPrinting
Paragraph	IsStyleSeparator
Revision	FormatDescription
Range	EndnoteOptions , FootnoteOptions , HTMLDivisions , SmartTags
Selection	ChildShapeRange , EndnoteOptions , FootnoteOptions , HasChildShapeRange , HTMLDivisions , SmartTags
Shape	CanvasItems , Child , Diagram , DiagramNode , HasDiagram , HasDiagramNode , ParentGroup
ShapeRange	CanvasItems , Child , Diagram , DiagramNode , HasDiagram , HasDiagramNode , ParentGroup
SmartTag	DownloadURL , Properties , XML
Style	LinkStyle , NoSpaceBetweenParagraphsOfSameStyle , Table
StyleSheet	Title
Table	ApplyStyleFirstColumn , ApplyStyleHeadingRows , ApplyStyleLastColumn , ApplyStyleLastRow
TableStyle	AllowBreakAcrossPage , ColumnStripe , RowStripe DisplayPageBoundaries , DisplaySmartTags , Reviewers , RevisionsBalloonShowConnectingLines , RevisionsBalloonSide , RevisionsBalloonWidth , RevisionsBalloonWidthType , RevisionsMode , RevisionsView , ShowComments , ShowFormatChanges , ShowInsertionsAndDeletions , ShowRevisionsAndComments
View	
WebOptions	TargetBrowser



New Properties (Alphabetic List)

Properties that have been added to existing objects in Microsoft Word 2002 are listed in the following table (sorted alphabetically).

New Property	Object(s)
<u>AllowBreakAcrossPage</u>	TableStyle
<u>ApplyStyleFirstColumn</u>	Table
<u>ApplyStyleHeadingRows</u>	Table
<u>ApplyStyleLastColumn</u>	Table
<u>ApplyStyleLastRow</u>	Table
<u>AutoCorrectEmail</u>	Application, Global
<u>AutoCreateNewDrawings</u>	Options
<u>AutoFormat</u>	Diagram
<u>AutoLayout</u>	Diagram
<u>AutomationSecurity</u>	Application
<u>BackgroundOpen</u>	Options
<u>Black</u>	ColorFormat
<u>BookFoldPrinting</u>	PageSetup
<u>BookFoldPrintingSheets</u>	PageSetup
<u>BookFoldRevPrinting</u>	PageSetup
<u>CanvasItems</u>	Shape, ShapeRange
<u>Child</u>	Shape, ShapeRange
<u>Children</u>	DiagramNode
<u>ChildShapeRange</u>	Selection
<u>ColumnStripe</u>	TableStyle
<u>CommentsColor</u>	Options
<u>CorrectTableCells</u>	AutoCorrect
<u>CtrlClickHyperlinkToOpen</u>	Options
<u>Cyan</u>	ColorFormat

<u>DataFieldIndex</u>	MappedDataField
<u>DataFieldName</u>	MappedDataField
<u>DefaultEPostageApp</u>	Options
<u>DefaultLegalBlackline</u>	Application
<u>DefaultTableStyle</u>	Document
<u>DefaultTargetFrame</u>	Document
<u>DefaultTextEncoding</u>	Options
<u>Diagram</u>	DiagramNode, Shape, ShapeRange
<u>DiagramNode</u>	Shape, ShapeRange
<u>DisableFeatures</u>	Document
<u>DisableFeaturesByDefault</u>	Options
<u>DisableFeaturesIntroducedAfter</u>	Document
<u>DisableFeaturesIntroducedAfterByDefault</u>	Options
<u>DisplayAutoCorrectOptions</u>	AutoCorrect
<u>DisplayPageBoundaries</u>	View
<u>DisplayPasteOptions</u>	Options
<u>DisplaySmartTagButtons</u>	Options
<u>DisplaySmartTags</u>	View
<u>DoNotEmbedSystemFonts</u>	Document
<u>DownloadURL</u>	SmartTag
<u>EmailSignatureEntries</u>	EmailSignature
<u>EmailTemplate</u>	Application
<u>EmbedLinguisticData</u>	Document
<u>EmbedSmartTag</u>	Document
<u>EmbedSmartTags</u>	EmailOptions
<u>EndnoteOptions</u>	Range, Selection
<u>FileDialog</u>	Application
<u>FirstChild</u>	DiagramNodeChildren
<u>FootnoteOptions</u>	Range, Selection
<u>FormatDescription</u>	Revision
<u>FormatScanning</u>	Options
<u>FormattingShowClear</u>	Document

<u>FormattingShowFilter</u>	Document
<u>FormattingShowFont</u>	Document
<u>FormattingShowNumbering</u>	Document
<u>FormattingShowParagraph</u>	Document
<u>HasChildShapeRange</u>	Selection
<u>HasDiagram</u>	Shape, ShapeRange
<u>HasDiagramNode</u>	Shape, ShapeRange
<u>HighlightMergeFields</u>	MailMerge
<u>HTMLDivisions</u>	Document, HTMLDivision, Range, Selection
<u>HTMLFidelity</u>	EmailOptions
<u>Included</u>	MailMergeDataSource
<u>Ink</u>	ColorFormat
<u>InsetPen</u>	LineFormat
<u>InvalidAddress</u>	MailMergeDataSource
<u>InvalidComments</u>	MailMergeDataSource
<u>IsPictureBullet</u>	InlineShape
<u>IsStyleSeparator</u>	Paragraph
<u>LabelSmartTags</u>	Options
<u>LastChild</u>	DiagramNodeChildren
<u>Layout</u>	DiagramNode
<u>LinkStyle</u>	Style
<u>ListPictureBullet</u>	ListPictureBullet
<u>LocalNetworkFile</u>	Options
<u>Magenta</u>	ColorFormat
<u>MailEnvelope</u>	Document
<u>MailFormat</u>	MailMerge
<u>MappedDataFields</u>	MailMergeDataSource
<u>NewColorOnReply</u>	EmailOptions
<u>NewDocument</u>	Application
<u>NoSpaceBetweenParagraphsOfSameStyle</u>	Style
<u>OverPrint</u>	ColorFormat
<u>ParentGroup</u>	Shape, ShapeRange

<u>PasswordEncryptionAlgorithm</u>	Document
<u>PasswordEncryptionFileProperties</u>	Document
<u>PasswordEncryptionKeyLength</u>	Document
<u>PasswordEncryptionProvider</u>	Document
<u>PasteAdjustParagraphSpacing</u>	Options
<u>PasteAdjustTableFormatting</u>	Options
<u>PasteAdjustWordSpacing</u>	Options
<u>PasteMergeFromPPT</u>	Options
<u>PasteMergeFromXL</u>	Options
<u>PasteMergeLists</u>	Options
<u>PasteSmartCutPaste</u>	Options
<u>PasteSmartStyleBehavior</u>	Options
<u>PictureBullet</u>	ListLevel
<u>PictureWrapType</u>	Options
<u>PlainTextStyle</u>	EmailOptions
<u>PreserveFormattingOnUpdate</u>	OLEFormat
<u>PromptUpdateStyle</u>	Options
<u>Properties</u>	SmartTag
<u>RecipientNameFromLeft</u>	Envelope
<u>RecipientNameFromTop</u>	Envelope
<u>RecipientPostalFromLeft</u>	Envelope
<u>RecipientPostalFromTop</u>	Envelope
<u>RecordCount</u>	MailMergeDataSource
<u>RemovePersonalInformation</u>	Document
<u>Reverse</u>	Diagram
<u>Reviewers</u>	View
<u>RevisionsBalloonPrintOrientation</u>	Options
<u>RevisionsBalloonShowConnectingLines</u>	View
<u>RevisionsBalloonSide</u>	View
<u>RevisionsBalloonWidth</u>	View
<u>RevisionsBalloonWidthType</u>	View
<u>RevisionsMode</u>	View
<u>RevisionsView</u>	View

<u>Root</u>	DiagramNode
<u>RowStripe</u>	TableStyle
<u>SaveNewWebPagesAsWebArchives</u>	DefaultWebOptions
<u>SenderNameFromLeft</u>	Envelope
<u>SenderNameFromTop</u>	Envelope
<u>SenderPostalFromLeft</u>	Envelope
<u>SenderPostalFromTop</u>	Envelope
<u>SequenceCheck</u>	Options
<u>ShowComments</u>	View
<u>ShowFormatChanges</u>	View
<u>ShowFormatError</u>	Options
<u>ShowInsertionsAndDeletions</u>	View
<u>ShowRevisionsAndComments</u>	View
<u>ShowSendToCustom</u>	MailMerge
<u>ShowStartupDialog</u>	Application
<u>ShowWindowsInTaskbar</u>	Application
<u>Signatures</u>	Document
<u>SmartParaSelection</u>	Options
<u>SmartTags</u>	Document, Range, Selection
<u>SmartTagsAsXMLProps</u>	Document
<u>StoreRSIDOnSave</u>	Options
<u>StyleSheets</u>	Document
<u>Table</u>	Style
<u>TableName</u>	MailMergeDataSource
<u>TargetBrowser</u>	DefaultWebOptions, WebOptions
<u>TaskPanels</u>	Application
<u>TextEncoding</u>	Document
<u>TextLineEnding</u>	Document
<u>TextShape</u>	DiagramNode
<u>TintAndShade</u>	ColorFormat
<u>Title</u>	StyleSheet

TypeNReplace

UseThemeStyleOnReply

Vertical

WarnBeforeSavingPrintingSendingMarkup

WizardState

XML

Yellow

Options

EmailOptions

Envelope, MailingLabel

Options

MailMerge

SmartTag

ColorFormat



New Methods (by Object)

Methods that have been added to existing objects in Microsoft Word 2002 are listed in the following table (sorted by object name).

Object	New Methods
CanvasShapes	<u>AddConnector</u>
ColorFormat	<u>SetCMYK</u>
DiagramNode	<u>AddNode</u> , <u>CloneNode</u> , <u>MoveNode</u> , <u>NextNode</u> , <u>PrevNode</u> , <u>ReplaceNode</u> , <u>SwapNode</u> , <u>TransferChildren</u>
DiagramNodeChildren	<u>AddNode</u>
Document	<u>AcceptAllRevisionsShown</u> , <u>CanCheckIn</u> , <u>CheckIn</u> , <u>CheckNewSmartTags</u> , <u>ConvertVietDoc</u> , <u>DeleteAllComments</u> , <u>DeleteAllCommentsShown</u> , <u>EndReview</u> , <u>RecheckSmartTags</u> , <u>RejectAllRevisionsShown</u> , <u>RemoveSmartTags</u> , <u>ReplyWithChanges</u> , <u>ResetFormFields</u> , <u>SendForReview</u> , <u>SetDefaultTableStyle</u> , <u>SetPasswordEncryptionOptions</u>
Documents	<u>CanCheckOut</u> , <u>CheckOut</u> , <u>DiscardConflict</u> , <u>OfflineConflict</u>
Envelope	<u>Options</u>
HTMLDivision	<u>HTMLDivisionParent</u>
ListLevel	<u>ApplyPictureBullet</u>
MailingLabel	<u>LabelOptions</u>
MailMerge	<u>ShowWizard</u>
MailMergeDataSource	<u>SetAllErrorFlags</u> , <u>SetAllIncludedFlags</u>
Paragraph	<u>SelectNumber</u>
Paragraphs	<u>DecreaseSpacing</u> , <u>IncreaseSpacing</u> <u>InsertStyleSeparator</u> , <u>PasteAndFormat</u> , <u>PasteAppendTable</u> , <u>PasteExcelTable</u> ,
Selection	

	<u>ShrinkDiscontiguousSelection</u> , <u>ToggleCharacterCode</u>
Shape	<u>CanvasCropBottom</u> , <u>CanvasCropLeft</u> , <u>CanvasCropRight</u> , <u>CanvasCropTop</u>
ShapeRange	<u>CanvasCropBottom</u> , <u>CanvasCropLeft</u> , <u>CanvasCropRight</u> , <u>CanvasCropTop</u>
Shapes	<u>AddCanvas</u> , <u>AddConnector</u> , <u>AddDiagram</u>
TableStyle	<u>Condition</u>



New Methods (Alphabetic List)

Methods that have been added to existing objects in Microsoft Word 2002 are listed in the following table (sorted alphabetically).

New Method	Object
<u>AcceptAllRevisionsShown</u>	Document
<u>AddCanvas</u>	Shapes
<u>AddConnector</u>	CanvasShapes, Shapes
<u>AddDiagram</u>	Shapes
<u>AddNode</u>	DiagramNode, DiagramNodeChildren
<u>ApplyPictureBullet</u>	ListLevel
<u>CanCheckIn</u>	Document
<u>CanCheckOut</u>	Documents
<u>CanvasCropBottom</u>	Shape, ShapeRange
<u>CanvasCropLeft</u>	Shape, ShapeRange
<u>CanvasCropRight</u>	Shape, ShapeRange
<u>CanvasCropTop</u>	Shape, ShapeRange
<u>CheckIn</u>	Document
<u>CheckNewSmartTags</u>	Document
<u>CheckOut</u>	Documents
<u>CloneNode</u>	DiagramNode
<u>Condition</u>	TableStyle
<u>ConvertVietDoc</u>	Document
<u>DecreaseSpacing</u>	Paragraphs
<u>DeleteAllComments</u>	Document
<u>DeleteAllCommentsShown</u>	Document
<u>DiscardConflict</u>	Documents
<u>EndReview</u>	Document

<u>HTMLDivisionParent</u>	HTMLDivision
<u>IncreaseSpacing</u>	Paragraphs
<u>InsertStyleSeparator</u>	Selection
<u>LabelOptions</u>	MailingLabel
<u>MoveNode</u>	DiagramNode
<u>NextNode</u>	DiagramNode
<u>OfflineConflict</u>	Documents
<u>Options</u>	Envelope
<u>PasteAndFormat</u>	Selection
<u>PasteAppendTable</u>	Selection
<u>PasteExcelTable</u>	Selection
<u>PrevNode</u>	DiagramNode
<u>RecheckSmartTags</u>	Document
<u>RejectAllRevisionsShown</u>	Document
<u>RemoveSmartTags</u>	Document
<u>ReplaceNode</u>	DiagramNode
<u>ReplyWithChanges</u>	Document
<u>ResetFormFields</u>	Document
<u>SelectNumber</u>	Paragraph
<u>SendForReview</u>	Document
<u>SetAllErrorFlags</u>	MailMergeDataSource
<u>SetAllIncludedFlags</u>	MailMergeDataSource
<u>SetCMYK</u>	ColorFormat
<u>SetDefaultTableStyle</u>	Document
<u>SetPasswordEncryptionOptions</u>	Document
<u>ShowWizard</u>	MailMerge
<u>ShrinkDiscontiguousSelection</u>	Selection
<u>SwapNode</u>	DiagramNode
<u>ToggleCharacterCode</u>	Selection
<u>TransferChildren</u>	DiagramNode



New Events

Events that have been added to the [Application](#) object in Microsoft Word 2002 are given in the following alphabetic list.

[EPostageInsert](#)

[EPostagePropertyDialog](#)

[MailMergeAfterMerge](#)

[MailMergeAfterRecordMerge](#)

[MailMergeBeforeMerge](#)

[MailMergeBeforeRecordMerge](#)

[MailMergeDataSourceLoad](#)

[MailMergeDataSourceValidate](#)

[MailMergeWizardSendToCustom](#)

[MailMergeWizardStateChange](#)

[WindowSize](#)



↳ [Show All](#)

Language-Specific Properties and Methods

The Microsoft Word 2002 Visual Basic object model has language-specific keywords for use with Asian and right-to-left languages. The availability of these language-specific keywords depends on the language support you have selected or installed. This topic includes information on the following Visual Basic language-specific items:

- [Properties](#)
- [Methods](#)
- [Arguments](#)

Language-Specific Properties

Properties that are available only in Asian or right-to-left languages are listed in the following table.

Property	Object
<u>AddBiDirectionalMarksWhenSavingTextFile</u>	Options
<u>AddControlCharacters</u>	Options
<u>AddHebDoubleQuote</u>	Options
<u>AddSpaceBetweenFarEastAndAlpha</u>	Paragraph, ParagraphForm Paragraphs
<u>AddSpaceBetweenFarEastAndDigit</u>	Paragraph, ParagraphForm Paragraphs
<u>AllowCombinedAuxiliaryForms</u>	Options
<u>AllowCompoundNounProcessing</u>	Options
<u>ApplyFarEastFontsToAscii</u>	Options
<u>ArabicMode</u>	Options
<u>ArabicNumeral</u>	Options
<u>AutoAdjustRightIndent</u>	Paragraph, ParagraphForm Paragraphs
<u>AutoFormatApplyFirstIndents</u>	Options
<u>AutoFormatAsYouTypeApplyClosings</u>	Options
<u>AutoFormatAsYouTypeApplyDates</u>	Options
<u>AutoFormatAsYouTypeApplyFirstIndents</u>	Options
<u>AutoFormatAsYouTypeAutoLetterWizard</u>	Options
<u>AutoFormatAsYouTypeDeleteAutoSpaces</u>	Options
<u>AutoFormatAsYouTypeInsertClosings</u>	Options
<u>AutoFormatAsYouTypeInsertOvers</u>	Options
<u>AutoFormatAsYouTypeMatchParentheses</u>	Options
<u>AutoFormatAsYouTypeReplaceFarEastDashes</u>	Options
<u>AutoFormatDeleteAutoSpaces</u>	Options
<u>AutoFormatMatchParentheses</u>	Options
<u>AutoFormatReplaceFarEastDashes</u>	Options

<u>BaseLineAlignment</u>	Paragraph, ParagraphForm Paragraphs
<u>BoldBi</u>	Font, Range
<u>BookFoldRevPrinting</u>	PageSetup
<u>BuiltinDictionary</u>	HangulHanjaConversionDic
<u>CharacterUnitFirstLineIndent</u>	Paragraph, ParagraphForm Paragraphs
<u>CharacterUnitLeftIndent</u>	Paragraph, ParagraphForm Paragraphs
<u>CharacterUnitRightIndent</u>	Paragraph, ParagraphForm Paragraphs
<u>CharacterWidth</u>	Range
<u>CharsLine</u>	PageSetup
<u>CheckHangulEndings</u>	Options
<u>ColorIndexBi</u>	Font
<u>CombineCharacters</u>	Range
<u>ConvertHighAnsiToFarEast</u>	Options
<u>CorrectHangulAndAlphabet</u>	AutoCorrect
<u>CorrectHangulEndings</u>	Find
<u>CursorMovement</u>	Options
<u>DiacriticColor</u>	Font
<u>DiacriticColorVal</u>	Options
<u>DisableCharacterSpaceGrid</u>	Font, Range
<u>DisableLineHeightGrid</u>	Paragraph, ParagraphForm Paragraphs
<u>DisplayGridLines</u>	Options
<u>DisplayLeftScrollBar</u>	Window
<u>DisplayRightRuler</u>	Window
<u>DocumentViewDirection</u>	Options
<u>DoNotEmbedSystemFonts</u>	Document
<u>DoubleQuote</u>	PageNumbers
<u>EmphasisMark</u>	Font, Range
<u>EnableHangulHanjaRecentOrdering</u>	Options Paragraph, ParagraphForm

<u>FarEastLineBreakControl</u>	Paragraphs
<u>FarEastLineBreakLanguage</u>	Document, Template
<u>FarEastLineBreakLevel</u>	Document, Template
<u>Filter</u>	Index
<u>FitTextWidth</u>	Range, Selection
<u>FlowDirection</u>	TextColumns
<u>GridOriginFromMargin</u>	Document
<u>GridSpaceBetweenHorizontalLines</u>	Document
<u>GridSpaceBetweenVerticalLines</u>	Document
<u>GutterPos</u>	PageSetup
<u>GutterStyle</u>	PageSetup
<u>HalfWidthPunctuationOnTopOfLine</u>	Paragraph, ParagraphForm
<u>HangingPunctuation</u>	Paragraphs
<u>HangulAndAlphabetAutoAdd</u>	Paragraph, ParagraphForm
<u>HangulAndAlphabetExceptions</u>	Paragraphs
<u>HangulHanjaDictionaries</u>	AutoCorrect
<u>HangulHanjaFastConversion</u>	AutoCorrect
<u>HebrewMode</u>	Application, Global
<u>HorizontalInVertical</u>	Options
<u>IMEAutomaticControl</u>	Options
<u>IMEMode</u>	Range
<u>InlineConversion</u>	Options
<u>ItalicBi</u>	Window
<u>JustificationMode</u>	Options
<u>Kana</u>	Font, Range
<u>KerningByAlgorithm</u>	Document, Template
<u>LayoutMode</u>	Range
<u>LinesPage</u>	Document, Template
<u>LineUnitAfter</u>	PageSetup
<u>LineUnitBefore</u>	PageSetup
	Paragraph, ParagraphForm
	Paragraphs
	Paragraph, ParagraphForm

<u>MatchAlefHamza</u>	Paragraphs
<u>MatchByte</u>	Find
<u>MatchControl</u>	Find
<u>MatchDiacritics</u>	Find
<u>MatchFuzzy</u>	Find
<u>MatchFuzzyAY</u>	Options
<u>MatchFuzzyBV</u>	Options
<u>MatchFuzzyByte</u>	Options
<u>MatchFuzzyCase</u>	Options
<u>MatchFuzzyDash</u>	Options
<u>MatchFuzzyDZ</u>	Options
<u>MatchFuzzyHF</u>	Options
<u>MatchFuzzyHiragana</u>	Options
<u>MatchFuzzyIterationMark</u>	Options
<u>MatchFuzzyKanji</u>	Options
<u>MatchFuzzyKiKu</u>	Options
<u>MatchFuzzyOldKana</u>	Options
<u>MatchFuzzyProlongedSoundMark</u>	Options
<u>MatchFuzzyPunctuation</u>	Options
<u>MatchFuzzySmallKana</u>	Options
<u>MatchFuzzySpace</u>	Options
<u>MatchFuzzyTC</u>	Options
<u>MatchFuzzyZJ</u>	Options
<u>MatchKashida</u>	Find
<u>MonthNames</u>	Options
<u>MultipleWordConversionsMode</u>	Options
<u>NameBi</u>	Font
<u>NoLineBreakAfter</u>	Document, Template
<u>NoLineBreakBefore</u>	Document, Template
<u>PrintEvenPagesInAscendingOrder</u>	Options
<u>PrintOddPagesInAscendingOrder</u>	Options
<u>ReadingOrder</u>	Paragraph, ParagraphForm

<u>RecipientNamefromLeft</u>	Paragraphs
<u>RecipientNamefromTop</u>	Envelope
<u>RecipientPostalfromLeft</u>	Envelope
<u>RecipientPostalfromTop</u>	Envelope
<u>SectionDirection</u>	PageSetup
<u>SenderNamefromLeft</u>	Envelope
<u>SenderNamefromTop</u>	Envelope
<u>SenderPostalfromLeft</u>	Envelope
<u>SenderPostalfromTop</u>	Envelope
<u>ShowControlCharacters</u>	Options
<u>ShowDiacritics</u>	Options
<u>ShowOptionalBreaks</u>	View
<u>SizeBi</u>	Font
<u>SortBy</u>	Index
<u>StrictFinalYaa</u>	Options
<u>StrictInitialAlefHamza</u>	Options
<u>TableDirection</u>	Rows, Table
<u>TwoLinesInOne</u>	Range
<u>TwoPagesOnOne</u>	PageSetup
<u>UseCharacterUnit</u>	Options
<u>UseDiffDiacColor</u>	Options
<u>UseGermanSpellingReform</u>	Options
<u>Vertical</u>	Envelope and MailingLabel
<u>VisualSelection</u>	Options

Language-Specific Methods

Methods that are available only in Asian or right-to-left languages are listed in the following table.

Method	Object
<u>BoldRun</u>	Selection
<u>CheckConsistency</u>	Document
<u>ClearAllFuzzyOptions</u>	Find
<u>ConvertHangulAndHanja</u>	Range
<u>ConvertVietDoc</u>	Document
<u>IndentCharWidth</u>	Paragraph, ParagraphFormat, Paragraphs
<u>IndentFirstLineCharWidth</u>	Paragraph, ParagraphFormat, Paragraphs
<u>ItalicRun</u>	Selection
<u>KeyboardBidi</u>	Application
<u>KeyboardLatin</u>	Application
<u>LtrPara</u>	Selection
<u>LtrRun</u>	Selection
<u>ModifyEnclosure</u>	Range
<u>PhoneticGuide</u>	Range
<u>RtlPara</u>	Selection
<u>RtlRun</u>	Selection
<u>SetAllFuzzyOptions</u>	Find
<u>TCSCConverter</u>	Range
<u>ToggleKeyboard</u>	Application

Language-Specific Arguments

Methods that are available in all languages, but that have one or more language-specific arguments, are listed in the following table.

Method	Object
Execute	Find
InsertDateTime	Range, Selection
InsertSymbol	Range, Selection
PrintOut	Application, Document, Window
Sort	Column, Range, Selection, Table

Note In order to [enable](#) the language-specific features in Microsoft Word, you must be running a 32-bit version of the Microsoft Windows operating system that has support for that particular language — for example, the Arabic version of Microsoft Windows 95 has [right-to-left](#) support. For more information, see [Learn about requirements for right-to-left or Asian languages](#).



Hidden Properties and Methods

In Microsoft Word 2002, the following properties and methods have been hidden. Hidden properties are supported only for backward compatibility.

Properties

Property	View	Object
BrowseToWindow	View	
EnlargeFontsLessThan	View	

Methods

Method	Object
UseAddressBook	MailMerge

↳ [Show All](#)

Returning an Object from a Collection

The [Item](#) method returns a single object from a collection. The following example sets the `firstDoc` variable to a [Document](#) object that represents the first document in the [Documents](#) collection.

```
Sub SetFirstDoc()  
    Dim docFirst As Document  
    Set docFirst = Documents.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 SetFirstDoc()  
    Dim docFirst As Document  
    Set docFirst = Documents(1)  
End Sub
```

Named Objects

Although you can usually specify an integer value with the **Item** method, it may be more convenient to return an object by name. The following example switches the focus to a document named Sales.doc.

```
Sub ActivateDocument()  
    Documents("Sales.doc").Activate  
    MsgBox ActiveDocument.Name  
End Sub
```

The following example selects the text marked by the first bookmark in the active document.

```
Sub SelectBookmark()  
    ActiveDocument.Bookmarks(1).Select  
    MsgBox Selection.Text  
End Sub
```

Not all collections can be indexed by name. To determine the valid collection index values, see the collection object topic.

Predefined Index Values

Some collections have predefined index values you can use to return single objects. Each predefined index value is represented by a constant. For example, you specify an **WdBorderType** constant with the **Borders** property to return a single **Border** object.

The following example adds a single 0.75 point border below the first paragraph in the selection.

```
Sub AddBorderToFirstParagraphInSelection()  
    With Selection.Paragraphs(1).Borders(wdBorderBottom)  
        .LineStyle = wdLineStyleSingle  
        .LineWidth = wdLineWidth300pt  
        .Color = wdColorBlue  
    End With  
End Sub
```



Converting WordBasic Macros to Visual Basic

Microsoft Word 2002 automatically converts the macros in a Word 6.x or Word 95 template the first time you do any of the following:

- Open the template
- Create a new document based on the template
- Attach the template to a document by using the **Templates** command (**Tools** menu).

A message is displayed on the status bar while the macros are being converted. After the conversion is complete, you must save the template to save the converted macros. If you don't save the template, Word converts the macros again the next time you use the template.

Note Word 2002 cannot convert Word 2.x macros directly. Instead, you need to open and save your Word 2.x templates in Word 6.x or Word 95 and then open them in Word 2002.

The conversion process converts each macro to a Visual Basic module. To see the converted macros, point to **Macro** on the **Tools** menu and click **Macros**. The macro names in the **Macros** dialog box appear as *macroname*.Main, where Main refers to the main subroutine in the converted macro (the subroutine that began with Sub MAIN in earlier versions of Word). To edit the converted macro, select a macro name and click **Edit** to display the Visual Basic module in the Visual Basic Editor.

Each WordBasic statement is modified to work with Visual Basic for Applications. The converted WordBasic macros are functionally equivalent to new Visual Basic for Applications macros you might write or record, but they are not identical. The following example is a WordBasic macro in a Word 95 template.

```
Sub MAIN
FormatFont .Name = "Arial", .Points = 10
Insert "Hello World"
End Sub
```

When the template is opened in Word, the macro is converted to the following code.

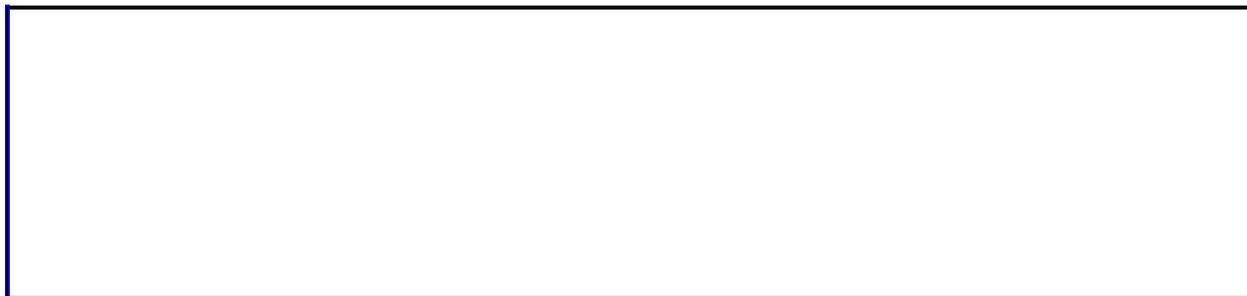
```
Public Sub Main()
WordBasic.FormatFont Font:="Arial", Points:=10
WordBasic.Insert "Hello World"
End Sub
```

Each statement in the converted macro begins with the [WordBasic](#) property. **WordBasic** is a property in the Word 2002 object model that returns an object with all the WordBasic statements and functions; this object makes it possible to run WordBasic macros in Word 2002.

Note If you save the template over the original template, the WordBasic macros will be permanently lost and previous versions of Word will not be able to use the converted macros.

The following Visual Basic macro is functionally the same as the preceding WordBasic macro, but doesn't use the **WordBasic** property.

```
Public Sub Main()
  With Selection.Font
    .Name = "Arial"
    .Size = 10
  End With
  Selection.TypeText Text:="Hello World"
End Sub
```



Visual Basic Equivalents for WordBasic Commands

A B C D E F G H I J K L M N O P R S T U V W Y

To find the Visual Basic property or method that's the equivalent of a WordBasic command, click the first letter of the WordBasic command name. Then scroll through the lists of WordBasic commands until you find the appropriate command. The right column includes sample Visual Basic syntax with jumps to topics in the Microsoft Word Visual Basic Help.

For information about converting macros, see [Converting WordBasic macros to Visual Basic](#).

For information about the differences between WordBasic and Visual Basic, see [Conceptual differences between WordBasic and Visual Basic](#).



Recording a Macro to Generate Code

If you are unsure of which Visual Basic method or property to use, you can turn on the macro recorder and manually perform the action. The macro recorder translates your actions into Visual Basic code. After you've recorded your actions, you can modify the code to do exactly what you want. For example if you don't know what property or method to use to indent a paragraph, do the following:

1. On the **Tools** menu, point to **Macro**, and then click **Record New Macro**.
2. Change the default macro name if you'd like and click **OK** to start the recorder.
3. On the **Format** menu, choose **Paragraph**.
4. Change the left paragraph indent value and click **OK**.
5. Click the **Stop Recording** button on the **Stop Recording** toolbar.
6. On the **Tools** menu, point to **Macro**, and then click **Macros**.
7. Select the macro name from Step 2 and click the **Edit** button.

View the Visual Basic code to determine the property that corresponds to the left paragraph indent (the **LeftIndent** property). Position the insertion point within `LeftIndent` and press F1 or click the **Help** button. Within the topic, you can view examples and review the objects that support the **LeftIndent** property (click Applies To).

Remarks

Recorded macros use the [Selection](#) property to return the [Selection](#) object. For example, the following instruction indents the selected paragraphs by a half inch.

```
Sub IndentParagraph()  
    Selection.ParagraphFormat.LeftIndent = InchesToPoints(0.5)  
End Sub
```

You can, however, modify the recorded macro to work with [Range](#) objects. For information, see [Revising recorded Visual Basic macros](#).



Finding Out Which Property or Method to Use

You can use the macro recorder to find out what methods or properties you need to accomplish a task in Microsoft Word. The macro recorder is a tool that translates your actions into Visual Basic instructions. For example, if you turn on the macro recorder and open a document named Examples.doc, the macro recorder records an instruction similar to the following.

```
Sub Macro1()  
'  
' Macro1 Macro  
' Macro recorded 9/22/2000 by JeffSmith  
'  
    Documents.Open FileName:="Examples.doc", ConfirmConversions:=Fal  
        ReadOnly:=False, AddToRecentFiles:=False, _  
        PasswordDocument:="", PasswordTemplate:="", _  
        Revert:=False, WritePasswordDocument:="", _  
        WritePasswordTemplate:="", Format:=wdOpenFormatAuto  
End Sub
```

The [Documents](#) property returns the [Documents](#) collection and the [Open](#) method opens the specified file name. When you're first learning Visual Basic, using the macro recorder will help you learn which properties and methods you need to use to accomplish a task.

For more information, see the following:

[Revising recorded Visual Basic macros](#)



Returning a Single Object from a Collection

Information about returning a single object is available in the object topic itself and in the collection object topic for the collection that contains the object. Most object topics include information about returning a single object as well as adding an object to the collection of those objects. Most collection object topics include information about returning the collection itself and adding an object to the collection.

To browse the Help topics for the objects and collections in Microsoft Word, see [Microsoft Word Objects](#).



↳ [Show All](#)

Working with Range Objects

A common task when using Visual Basic is to specify an area in a document and then do something with it, such as insert text or apply formatting. For example, you may want to write a macro that locates a word or phrase within a portion of a document. The portion of the document can be represented by a **Range** object. After the **Range** object is identified, methods and properties of the **Range** object can be applied in order to modify the contents of the range.

A **Range** object refers to a contiguous area in a document. Each **Range** object is defined by a starting and ending character position. Similar to the way bookmarks are used in a document, **Range** objects are used in Visual Basic procedures to identify specific portions of a document. A **Range** object can be as small as the insertion point or as large as the entire document. However, unlike a bookmark, a **Range** object only exists while the procedure that defined it is running.

The **Start**, **End** and **StoryType** properties uniquely identify a **Range** object. The **Start** and **End** properties return or set the starting and ending character positions of the **Range** object. The character position at the beginning of the document is zero, the position after the first character is one, and so on. There are eleven different story types represented by the **WdStoryType** constants of the **StoryType** property.

Note **Range** objects are independent of the selection. That is, you can define and modify a range without changing the current selection. You can also define multiple ranges in a document, while there is only one selection per document pane.

Using the Range method

The [Range](#) method is used to create a **Range** object in the specified document. The **Range** method (which is available from the [Document](#) object) returns a **Range** object located in the main [story](#) given a start and end point. The following example creates a **Range** object that is assigned to a variable.

```
Sub SetNewRange()  
    Dim rngDoc As Range  
    Set rngDoc = ActiveDocument.Range(Start:=0, End:=10)  
End Sub
```

The variable refers to the first ten characters in the active document. You can see that the **Range** object has been created when you apply a property or method to the **Range** object stored in a variable. The following example applies bold formatting to the first ten characters in the active document.

```
Sub SetBoldRange()  
    Dim rngDoc As Range  
    Set rngDoc = ActiveDocument.Range(Start:=0, End:=10)  
    rngDoc.Bold = True  
End Sub
```

When you need to refer to a **Range** object multiple times, you can use the **Set** statement to set a variable equal to the **Range** object. However, if you only need to perform a single action on a **Range** object, there's no need to store the object in a variable. The same results can be achieved using just one instruction that identifies the range and changes the [Bold](#) property.

```
Sub BoldRange()  
    ActiveDocument.Range(Start:=0, End:=10).Bold = True  
End Sub
```

Like a bookmark, a range can span a group of characters or mark a location in a document. The **Range** object in the following example has the same starting and ending points. The range does not include any text. The following example inserts text at the beginning of the active document.

```
Sub InsertTextBeforeRange()  
    Dim rngDoc As Range  
    Set rngDoc = ActiveDocument.Range(Start:=0, End:=0)
```

```
rngDoc.InsertBefore "Hello "  
End Sub
```

You can define the beginning and end points of a range using the character position numbers as shown above, or use the **Start** and **End** properties with objects such as [Selection](#), [Bookmark](#), or **Range**. The following example creates a **Range** object beginning at the start of the second paragraph and ending after the third paragraph.

```
Sub NewRange()  
    Dim doc As Document  
    Dim rngDoc As Range  
  
    Set doc = ActiveDocument  
    Set rngDoc = doc.Range(Start:=doc.Paragraphs(2).Range.Start, _  
        End:=doc.Paragraphs(3).Range.End)  
End Sub
```

For additional information and examples, see the [Range](#) method.

Using the Range property

The **Range** property appears on multiple objects, such as [Paragraph](#), [Bookmark](#), and [Cell](#), and is used to return a **Range** object. The following example returns a **Range** object that refers to the first paragraph in the active document.

```
Sub SetParagraphRange()  
    Dim rngParagraph As Range  
    Set rngParagraph = ActiveDocument.Paragraphs(1).Range  
End Sub
```

After you have a **Range** object, you can use any of its properties or methods to modify the **Range** object. The following example selects the second paragraph in the active document and then centers the selection.

```
Sub FormatRange()  
    ActiveDocument.Paragraphs(2).Range.Select  
    Selection.ParagraphFormat.Alignment = wdAlignParagraphJustify  
End Sub
```

If you need to apply numerous properties or methods to the same **Range** object, you can use the **With...End With** structure. The following example formats the text in the first paragraph of the active document.

```
Sub FormatFirstParagraph()  
    Dim rngParagraph As Range  
    Set rngParagraph = ActiveDocument.Paragraphs(1).Range  
    With rngParagraph  
        .Bold = True  
        .ParagraphFormat.Alignment = wdAlignParagraphCenter  
        With .Font  
            .Name = "Stencil"  
            .Size = 15  
        End With  
    End With  
End Sub
```

For additional information and examples, see the [Range](#) property topic.

Redefining a Range object

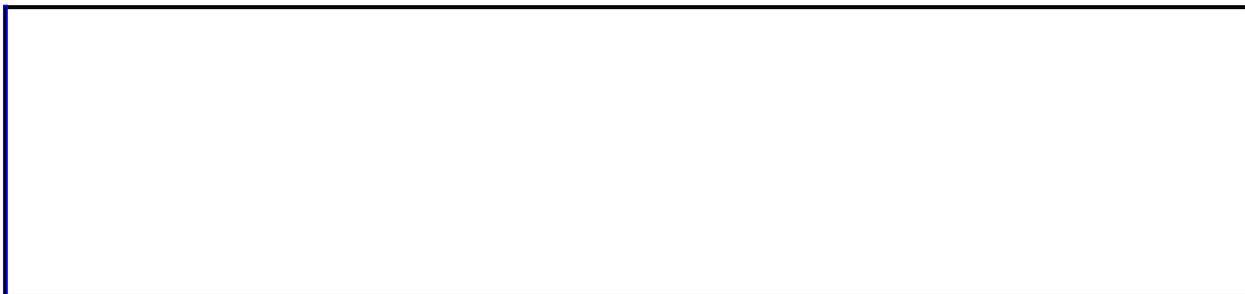
Use the [SetRange](#) method to redefine an existing **Range** object. The following example defines a range as the current selection. The **SetRange** method then redefines the range so that it refers to current selection plus the next ten characters.

```
Sub ExpandRange()  
    Dim rngParagraph As Range  
    Set rngParagraph = Selection.Range  
    rngParagraph.SetRange Start:=rngParagraph.Start, _  
        End:=rngParagraph.End + 10  
End Sub
```

For additional information and examples, see the **SetRange** method.

Note When debugging your macros, you can use the [Select](#) method to ensure that a **Range** object is referring to the correct range of text. For example, the following example selects a **Range** object, which refers the second and third paragraphs in the active document, and then formats the font of the selection.

```
Sub SelectRange()  
    Dim rngParagraph As Range  
  
    Set rngParagraph = ActiveDocument.Paragraphs(2).Range  
  
    rngParagraph.SetRange Start:=rngParagraph.Start, _  
        End:=ActiveDocument.Paragraphs(3).Range.End  
    rngParagraph.Select  
  
    Selection.Font.Italic = True  
End Sub
```



Modifying a Portion of a Document

Visual Basic includes objects which you can use to modify the following document elements: characters, words, sentences, paragraphs and sections. The following table includes the properties that correspond to these document elements and the objects they return.

This expression	Returns this object
<u>Words(index)</u>	<u>Range</u>
<u>Characters(index)</u>	<u>Range</u>
<u>Sentences(index)</u>	<u>Range</u>
<u>Paragraphs(index)</u>	<u>Paragraph</u>
<u>Sections(index)</u>	<u>Section</u>

When these properties are used without an index, a collection object with the same name is returned. For example, the **Paragraphs** property returns the [Paragraphs](#) collection object. However, if you identify an item within these collections by index, the object in the second column of the table is returned. For example, `words(1)` returns a **Range** object. After you have a **Range** object, you can use any of the range properties or methods to modify the **Range** object. For example, the following instruction copies the first word in the selection to the Clipboard.

```
Sub CopyWord()  
    Selection.Words(1).Copy  
End Sub
```

Note The items in the [Paragraphs](#) and [Sections](#) collections are singular forms of the collection rather than **Range** objects. However, the **Range** property (which returns a **Range** object) is available from both the **Paragraph** and **Section** objects. For example, the following instruction copies the first paragraph in the active document to the Clipboard.

```
Sub CopyParagraph()  
    ActiveDocument.Paragraphs(1).Range.Copy
```

End Sub

All of the document element properties in the preceding table are available from the **Document**, **Selection**, and **Range** objects. The following examples demonstrate how you can drill down to these properties from [Document](#), [Selection](#), and **Range** objects.

The following example sets the case of the first word in the active document.

```
Sub ChangeCase()  
    ActiveDocument.Words(1).Case = wdUpperCase  
End Sub
```

The following example sets the bottom margin of the current section to 0.5 inch.

```
Sub ChangeSectionMargin()  
    Selection.Sections(1).PageSetup.BottomMargin = InchesToPoints(0.5)  
End Sub
```

The following example double spaces the text in the active document (the [Content](#) property returns a **Range** object).

```
Sub DoubleSpaceDocument()  
    ActiveDocument.Content.ParagraphFormat.Space2  
End Sub
```

Modifying a group of document elements

To modify a range of text that consists of a group of document elements (characters, words, sentences, paragraphs or sections), you need to create a **Range** object. The **Range** method creates a **Range** object given a start and end point. For example, the following instruction creates a **Range** object that refers to the first ten characters in the active document.

```
Sub SetRangeForFirstTenCharacters()  
    Dim rngTenCharacters As Range  
    Set rngTenCharacters = ActiveDocument.Range(Start:=0, End:=10)  
End Sub
```

Using the **Start** and **End** properties with a **Range** object, you can create a new **Range** object that refers to a group of document elements. For example, the following instruction creates a **Range** object (myRange) that refers to the first three words in the active document.

```
Sub SetRangeForFirstThreeWords()  
    Dim docActive As Document  
    Dim rngThreeWords As Range  
    Set docActive = ActiveDocument  
    Set rngThreeWords = docActive.Range(Start:=docActive.Words(1).St  
        End:=docActive.Words(3).End)  
End Sub
```

The following example creates a **Range** object (aRange) beginning at the start of the second paragraph and ending after the third paragraph.

```
Sub SetParagraphRange()  
    Dim docActive As Document  
    Dim rngParagraphs As Range  
    Set docActive = ActiveDocument  
    Set rngParagraphs = docActive.Range(Start:=docActive.Paragraphs(  
        End:=docActive.Paragraphs(3).Range.End)  
End Sub
```

For more information on defining **Range** objects, see [Working with Range objects](#).



Object Doesn't Support this Property or Method

The "object doesn't support this property or method" error occurs when you try to use a method or property that the specified object doesn't support. For example, the following instruction results in an error.

```
ActiveDocument.Copy
```

The [ActiveDocument](#) property returns a [Document](#) object. A [Copy](#) method or property is not available for the **Document** object so an error occurs. To determine what properties and methods are available for an object, do any of the following.

- Use the Object Browser to determine what members (properties and methods) are available for the selected class (object).
- Use the Auto List Members feature in the Visual Basic Editor. When you type a period (.) after a property or method in the Visual Basic Editor, a list of available properties and methods is displayed
- Use Word Visual Basic Help to determine which properties and methods can be used with an object. Each object topic in Help includes a Properties and Methods jump that displays a list of properties and methods for the object. Press F1 in the Object Browser or a module to display the appropriate Help topic.
- Use the **TypeName** function to determine the type of object returned by an expression. The following example displays "Range" because the [Content](#) property returns a [Range](#) object.

```
MsgBox TypeName(ActiveDocument.Content)
```



Working with Document Objects

In Visual Basic, the methods for modifying files are methods of the [Document](#) object or the [Documents](#) collection object. This topic includes Visual Basic examples related to the following tasks:

- [Creating a new document](#)
- [Opening a document](#)
- [Saving an existing document](#)
- [Saving a new document](#)
- [Activating a document](#)
- [Determining if a document is open](#)
- [Referring to the active document](#)

Creating a new document

The **Documents** collection includes all of the open documents. To create a new document, use the [Add](#) method to add a **Document** object to the **Documents** collection. The following instruction creates a new document.

```
Documents.Add
```

A better way to create a new document is to assign the return value to an object variable. The **Add** method returns a **Document** object that refers to the new document. In the following example, the **Document** object returned by the **Add** method is assigned to an object variable. Then several properties and methods of the **Document** object are set. You can easily control the new document using an object variable.

```
Sub NewSampleDoc()  
    Dim docNew As Document  
    Set docNew = Documents.Add  
    With docNew  
        .Content.Font.Name = "Tahoma"  
        .SaveAs FileName:="Sample.doc"  
    End With  
End Sub
```

Opening a document

To open an existing document, use the [Open](#) method with the **Documents** collection. The following instruction opens a document named Sample.doc located in the MyFolder folder.

```
Sub OpenDocument()  
    Documents.Open FileName:="C:\MyFolder\Sample.doc"  
End Sub
```

Saving an existing document

To save a single document, use the [Save](#) method with the **Document** object. The following instruction saves the document named Sales.doc.

```
Sub SaveDocument()  
    Documents("Sales.doc").Save  
End Sub
```

You can save all open documents by applying the **Save** method to the **Documents** collection. The following instruction saves all open documents.

```
Sub SaveAllOpenDocuments()  
    Documents.Save  
End Sub
```

Saving a new document

To save a single document, use the [SaveAs](#) method with a **Document** object. The following instruction saves the active document as "Temp.doc" in the current folder.

```
Sub SaveNewDocument()  
    ActiveDocument.SaveAs FileName:="Temp.doc"  
End Sub
```

The ***FileName*** argument can include only the file name or the complete path (for example, "C:\Documents\Temporary File.doc").

Closing documents

To close a single document, use the [Close](#) method with a **Document** object. The following instruction closes and saves the document named Sales.doc.

```
Sub CloseDocument()  
    Documents("Sales.doc").Close SaveChanges:=wdSaveChanges  
End Sub
```

You can close all open documents by applying the **Close** method to the **Documents** collection. The following instruction closes all documents without saving changes.

```
Sub CloseAllDocuments()  
    Documents.Close SaveChanges:=wdDoNotSaveChanges  
End Sub
```

The following example prompts the user to save each document before the document is closed.

```
Sub PromptToSaveAndClose()  
    Dim doc As Document  
    For Each doc In Documents  
        doc.Close SaveChanges:=wdPromptToSaveChanges  
    Next  
End Sub
```

Activating a document

To change the active document, use the [Activate](#) method with a **Document** object. The following instruction activates the open document named Sales.doc.

```
Sub ActivateDocument()  
    Documents("Sales.doc").Activate  
End Sub
```

Determining if a document is open

To determine if a document is open, you can enumerate the **Documents** collection by using a **For Each...Next** statement. The following example activates the document named Sample.doc if the document is open, or opens Sample.doc if it's not currently open.

```
Sub ActivateOrOpenDocument()  
    Dim doc As Document  
    Dim docFound As Boolean  
  
    For Each doc In Documents  
        If InStr(1, doc.Name, "sample.doc", 1) Then  
            doc.Activate  
            docFound = True  
            Exit For  
        Else  
            docFound = False  
        End If  
    Next doc  
  
    If docFound = False Then Documents.Open FileName:="Sample.doc"  
End Sub
```

Referring to the active document

Instead of referring to a document by name or index number — for example `Documents("Sales.doc")` — the [ActiveDocument](#) property returns a **Document** object which refers to the active document (the document with the focus). The following example displays the name of the active document, or if there are no documents open, it displays a message.

```
Sub ActiveDocumentName()  
    If Documents.Count >= 1 Then  
        MsgBox ActiveDocument.Name  
    Else  
        MsgBox "No documents are open"  
    End If  
End Sub
```



Selecting Text in a Document

Use the [Select](#) method to select an item in a document. The **Select** method is available from several objects, such as [Bookmark](#), [Field](#), [Range](#), and [Table](#). The following example selects the first table in the active document.

```
Sub SelectTable()  
    ActiveDocument.Tables(1).Select  
End Sub
```

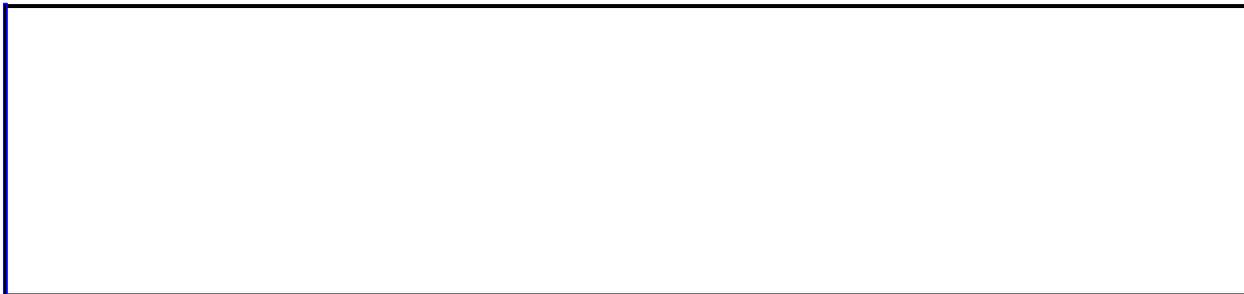
The following example selects the first field in the active document.

```
Sub SelectField()  
    ActiveDocument.Fields(1).Select  
End Sub
```

The following example selects the first four paragraphs in the active document. The [Range](#) method is used to create a **Range** object which refers to the first four paragraphs. The **Select** method is then applied to the **Range** object.

```
Sub SelectRange()  
    Dim rngParagraphs As Range  
    Set rngParagraphs = ActiveDocument.Range( _  
        Start:=ActiveDocument.Paragraphs(1).Range.Start, _  
        End:=ActiveDocument.Paragraphs(4).Range.End)  
    rngParagraphs.Select  
End Sub
```

For more information, see [Working with the Selection object](#).



Inserting Text in a Document

Use the [InsertAfter](#) or [InsertBefore](#) method to insert text before or after a [Selection](#) or [Range](#) object. The following example inserts text at the end of the active document.

```
Sub InsertTextAtEndOfDocument()  
    ActiveDocument.Content.InsertAfter Text:=" The end."  
End Sub
```

The following example inserts text before the selection.

```
Sub AddTextBeforeSelection()  
    Selection.InsertBefore Text:="new text "  
End Sub
```

After using the **InsertBefore** or **InsertAfter** method, the **Range** or **Selection** expands to include the new text. Use the [Collapse](#) method to collapse a **Selection** or **Range** to the beginning or ending point.



Requested Member of the Collection Does Not Exist

The "requested member of the collection does not exist" error occurs when you try to access an object that doesn't exist. For example, the following instruction may post an error if the active document doesn't contain at least one table.

```
Sub SelectTable()  
    ActiveDocument.Tables(1).Select  
End Sub
```

To avoid this error when accessing a member of a collection, ensure that the member exists prior to accessing the collection member. If you're accessing the member by index number, you can use the [Count](#) property to determine if the member exists. The following example selects the first table if there is at least one table in the active document.

```
Sub SelectFirstTable()  
    If ActiveDocument.Tables.Count > 0 Then  
        ActiveDocument.Tables(1).Select  
    Else  
        MsgBox "Document doesn't contain a table"  
    End If  
End Sub
```

If you're accessing a collection member by name, you can loop on the elements in a collection using a **For Each...Next** loop to determine if the named member is part of the collection. For example, the following example deletes the AutoCorrect entry named "acheive" if it's part of the [AutoCorrectEntries](#) collection. For more information, see [Looping Through a Collection](#).

```
Sub DeleteAutoTextEntry()  
    Dim aceEntry As AutoCorrectEntry  
    For Each aceEntry In AutoCorrect.Entries  
        If aceEntry.Name = "acheive" Then aceEntry.Delete  
    Next aceEntry  
End Sub
```



Looping Through a Collection

There are several different ways you can loop on the elements of a collection. However, the recommended method for looping on a collection is to use the **For Each...Next** loop. In this structure, Visual Basic repeats a block of statements for each object in a collection. The following example displays the name of each document in the [Documents](#) collection.

```
Sub LoopThroughOpenDocuments()  
    Dim docOpen As Document  
  
    For Each docOpen In Documents  
        MsgBox docOpen.Name  
    Next docOpen  
End Sub
```

Instead of displaying each element name in a message box, you can use an array to store the information. This example uses an array to store the name of each bookmark contained in the active document.

```
Sub LoopThroughBookmarks()  
    Dim bkMark As Bookmark  
    Dim strMarks() As String  
    Dim intCount As Integer  
  
    If ActiveDocument.Bookmarks.Count > 0 Then  
        ReDim strMarks(ActiveDocument.Bookmarks.Count - 1)  
        intCount = 0  
        For Each bkMark In ActiveDocument.Bookmarks  
            strMarks(intCount) = bkMark.Name  
            intCount = intCount + 1  
        Next bkMark  
    End If  
End Sub
```

You can loop through a collection to conditionally perform a task on members of the collection. For example, the following example updates the DATE fields in the active document.

```
Sub UpdateDateFields()  
    Dim fldDate As Field  
  
    For Each fldDate In ActiveDocument.Fields  
        If InStr(1, fldDate.Code, "Date", 1) Then fldDate.Update  
    Next fldDate  
End Sub
```

You can loop through a collection to determine if an element exists. For example, the following example displays a message if an AutoText entry named "Filename" is part of the [AutoTextEntries](#) collection.

```
Sub FindAutoTextEntry()  
    Dim atxtEntry As AutoTextEntry  
  
    For Each atxtEntry In ActiveDocument.AttachedTemplate.AutoTextEn  
        If atxtEntry.Name = "Filename" Then _  
            MsgBox "The Filename AutoText entry exists."  
    Next atxtEntry  
End Sub
```



Prompting for Information

There are several different ways you can prompt for information from a user.

- Use the Visual Basic **InputBox** function to display a dialog box with a message and an edit box. When the user clicks OK, the function returns the text that the user entered.
- Use the Visual Basic **MsgBox** function to display a simple message. You can display several different command buttons and icons. This function returns a number that indicates which button was clicked.
- Use a [built-in Microsoft Word dialog box](#) to get user input for a specific Word feature.
- Use a custom form to get user input. For information on adding controls to a form, see [Add a control to a form](#).



Returning Text from a Document

Use the [Text](#) property to return text from a **Range** or **Selection** object. The following example selects the next paragraph formatted with the Heading 1 style. The contents of the **Text** property are displayed by the **MsgBox** function.

```
Sub FindHeadingStyle()  
    With Selection.Find  
        .ClearFormatting  
        .Style = wdStyleHeading1  
        .Execute FindText:="", Format:=True, _  
            Forward:=True, Wrap:=wdFindStop  
        If .Found = True Then MsgBox Selection.Text  
    End With  
End Sub
```

The following instruction returns and displays the selected text.

```
Sub ShowSelection()  
    Dim strText As String  
    strText = Selection.Text  
    MsgBox strText  
End Sub
```

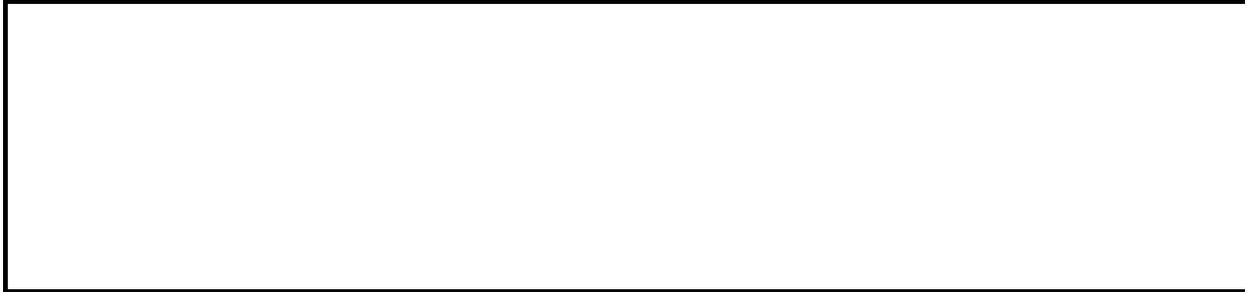
The following example returns the first word in the active document. Each item in the [Words](#) collection is a [Range](#) object that represents one word.

```
Sub ShowFirstWord()  
    Dim strFirstWord As String  
    strFirstWord = ActiveDocument.Words(1).Text  
    MsgBox strFirstWord  
End Sub
```

The following example returns the text associated with the first bookmark in the active document.

```
Sub ShowFirstBookmark()  
    Dim strBookmark As String  
    If ActiveDocument.Bookmarks.Count > 0 Then
```

```
        strBookmark = ActiveDocument.Bookmarks(1).Range.Text
    MsgBox strBookmark
End If
End Sub
```



Determining Whether the Application Property is Necessary

Many of the properties and methods of the [Application](#) object can be used without the **Application** object qualifier. For example the [ActiveDocument](#) property can be used without the **Application** object qualifier. Instead of writing `Application.ActiveDocument.PrintOut`, you can write `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.



Displaying Built-in Word Dialog Boxes

This topic contains the following information and examples:

- [Showing a built-in dialog box](#)
- [Returning and changing dialog box settings](#)
- [Checking how a dialog box was closed](#)

Showing a built-in dialog box

You can display a built-in dialog box to get user input or to control Microsoft Word using Visual Basic. The [Show](#) method of the **Dialog** object displays and executes any action taken in a built-in Word dialog box. To access a particular built-in Word dialog box, you specify a **WdWordDialog** constant with the [Dialogs](#) property. For example, the following macro instruction displays the **Open** dialog box (**wdDialogFileOpen**).

```
Sub ShowOpenDialog()  
    Dialogs(wdDialogFileOpen).Show  
End Sub
```

If a file is selected and **OK** is clicked, the file is opened (the action is executed). The following example displays the **Print** dialog box (**wdDialogFilePrint**).

```
Sub ShowPrintDialog()  
    Dialogs(wdDialogFilePrint).Show  
End Sub
```

Set the [DefaultTab](#) property to access a particular tab in a Word dialog box. The following example displays the **Page Border** tab in the **Borders and Shading** dialog box (**Format** menu).

```
Sub ShowBorderDialog()  
    With Dialogs(wdDialogFormatBordersAndShading)  
        .DefaultTab = wdDialogFormatBordersAndShadingTabPageBorder  
        .Show  
    End With  
End Sub
```

The [Display](#) method displays a dialog box without executing the actions taken in the dialog box. This can be useful if you want to prompt the user with a built-in dialog box and return the settings. For example, the following macro instruction displays the **User Information** tab from the **Options** dialog box (**Tools** menu) and then returns and displays the user name.

```
Sub DisplayUserInfoDialog()  
    With Dialogs(wdDialogToolsOptionsUserInfo)  
        .Display  
        MsgBox .Name  
    End With  
End Sub
```

```
End With
End Sub
```

Note You can also use Word's Visual Basic for Applications properties to display the user information without displaying the dialog box. The following example uses the [UserName](#) property for the [Application](#) object to display the user name for the application without displaying the **User Information** dialog box.

```
Sub DisplayUserInfo()
    MsgBox Application.UserName
End Sub
```

If the user name is changed in the previous example, the change is not set in the dialog box. Use the [Execute](#) method to execute the settings in a dialog box without displaying the dialog box. The following example displays the **User Information** dialog box, and if the name is not an empty string, the settings are set in the dialog box by using the **Execute** method.

```
Sub ShowAndSetUserInfoDialogBox()
    With Dialogs(wdDialogToolsOptionsUserInfo)
        .Display
        If .Name <> "" Then .Execute
    End With
End Sub
```

Note Use the VBA properties and methods in Word to set the user information without displaying the dialog box. The following code example changes the user name through the **UserName** property of the **Application** object, and then it displays the **User Information** dialog box to show that the change has been made. Note that displaying the dialog box is not necessary to change the value of a dialog box.

```
Sub SetUserName()
    Application.UserName = "Jeff Smith"
    Dialogs(wdDialogToolsOptionsUserInfo).Display
End Sub
```

Returning and changing dialog box settings

It's not very efficient to use a **Dialog** object to return or change a value for a dialog box when you can return or change it using a property or method. Also, in most, if not all, cases, when VBA code is used in place of accessing the Dialog object, code is simpler and shorter. Therefore, the following examples also include corresponding examples that use corresponding VBA properties to perform the same tasks.

Prior to returning or changing a dialog box setting using the **Dialog** object, you need to identify the individual dialog box. This is done by using the **Dialogs** property with a **WdWordDialog** constant. After you have instantiated a **Dialog** object you can return or set options in the dialog box. The following example displays the right indent from the **Paragraphs** dialog box.

```
Sub ShowRightIndent()  
    Dim dlgParagraph As Dialog  
    Set dlgParagraph = Dialogs(wdDialogFormatParagraph)  
    MsgBox "Right indent = " & dlgParagraph.RightIndent  
End Sub
```

Note You can use the VBA properties and methods of Word to display the right indent setting for the paragraph. The following example uses the **RightIndent** property of the **ParagraphFormat** object to display the right indent for the paragraph at the insertion point position.

```
Sub ShowRightIndexForSelectedParagraph()  
    MsgBox Selection.ParagraphFormat.RightIndent  
End Sub
```

Just as you can return dialog box settings, you can also set dialog box settings. The following example marks the **Keep with next** check box in the **Paragraph** dialog box.

```
Sub SetKeepWithNext()  
    With Dialogs(wdDialogFormatParagraph)  
        .KeepWithNext = 1  
        .Execute  
    End With  
End Sub
```

Note You can also use the VBA properties and methods to change the right indent for the paragraph. The following example uses the [KeepWithNext](#) property of the [ParagraphFormat](#) object to keep the selected paragraph with the following paragraph.

```
Sub SetKeepWithNextForSelectedParagraph()  
    Selection.ParagraphFormat.KeepWithNext = True  
End Sub
```

Note Use the [Update](#) method to ensure that the dialog box values reflect the current values. It may be necessary to use the **Update** method if you define a dialog box variable early in your macro and later want to return or change the current settings.

Checking how a dialog box was closed

The value returned by the **Show** and **Display** methods indicates which button was clicked to close the dialog box. The following example displays the **Break** dialog box, and if **OK** is clicked, a message is displayed on the status bar.

```
Sub DialogBoxButtons()  
    If Dialogs(wdDialogInsertBreak).Show = -1 Then  
        StatusBar = "Break inserted"  
    End If  
End Sub
```

The following table describes the return values associated with buttons in dialog boxes.

Return value	Description
-2	The Close button.
-1	The OK button.
0 (zero)	The Cancel button.
> 0 (zero)	A command button: 1 is the first button, 2 is the second button, and so on.



Error Accessing a Table Row or Column

When you try to access an individual row or column in a drawn table, a runtime error may occur if the table is not uniform. For example, the following instruction posts an error if the first table in the active document doesn't have the same number of rows in each column.

```
Sub RemoveTableBorders()  
    ActiveDocument.Tables(1).Rows(1).Borders.Enable = False  
End Sub
```

You can avoid this error by first selecting the cells in a column or row using the [SelectColumn](#) or [SelectRow](#) method. After the selection is made, use the [Cells](#) property with the [Selection](#) object. The following example selects the first row in the first document table. The **Cells** property is used to access the selected cells (all the cells in the first row) so that borders can be removed.

```
Sub RemoveTableBorders()  
    ActiveDocument.Tables(1).Cell(1, 1).Select  
    With Selection  
        .SelectRow  
        .Cells.Borders.Enable = False  
    End With  
End Sub
```

The following example selects the first column in the first document table. The **For Each...Next** loop is used to add text to each cell in the selection (all the cells in the first column).

```
Sub AddTextToTableCells()  
    Dim intCell As Integer  
    Dim oCell As Cell  
  
    ActiveDocument.Tables(1).Cell(1, 1).Select  
    Selection.SelectColumn  
    intCell = 1
```

```
For Each oCell In Selection.Cells
    oCell.Range.Text = "Cell " & intCell
    intCell = intCell + 1
Next oCell
End Sub
```



Applying Formatting to Text

This topic includes Visual Basic examples related to the following tasks:

- [Applying formatting to the selection](#)
- [Applying formatting to a range](#)
- [Inserting text and applying character and paragraph formatting](#)
- [Toggling the space before a paragraph between 12 points and none](#)
- [Toggling bold formatting](#)
- [Increasing the left margin by 0.5 inch](#)

Applying formatting to the selection

The following example uses the [Selection](#) property to apply character and paragraph formatting to the selected text. Use the [Font](#) property to access character formatting properties and methods and the [ParagraphFormat](#) property to access paragraph formatting properties and methods.

```
Sub FormatSelection()  
    With Selection.Font  
        .Name = "Times New Roman"  
        .Size = 14  
        .AllCaps = True  
    End With  
    With Selection.ParagraphFormat  
        .LeftIndent = InchesToPoints(0.5)  
        .Space1  
    End With  
End Sub
```

Applying formatting to a range

The following example defines a [Range](#) object that refers to the first three paragraphs in the active document. The **Range** is formatted by applying properties of the [Font](#) and [ParagraphFormat](#) objects.

```
Sub FormatRange()  
    Dim rngFormat As Range  
    Set rngFormat = ActiveDocument.Range( _  
        Start:=ActiveDocument.Paragraphs(1).Range.Start, _  
        End:=ActiveDocument.Paragraphs(3).Range.End)  
    With rngFormat  
        .Font.Name = "Arial"  
        .ParagraphFormat.Alignment = wdAlignParagraphJustify  
    End With  
End Sub
```

Inserting text and applying character and paragraph formatting

The following example adds the word Title at the top of the current document. The first paragraph is center aligned and a half inch space is added after the paragraph. The word Title is formatted with 24 point Arial font.

```
Sub InsertFormatText()  
    Dim rngFormat As Range  
    Set rngFormat = ActiveDocument.Range(Start:=0, End:=0)  
    With rngFormat  
        .InsertAfter Text:="Title"  
        .InsertParagraphAfter  
        With .Font  
            .Name = "Tahoma"  
            .Size = 24  
            .Bold = True  
        End With  
    End With  
    With ActiveDocument.Paragraphs(1)  
        .Alignment = wdAlignParagraphCenter  
        .SpaceAfter = InchesToPoints(0.5)  
    End With  
End Sub
```

Toggling the space before a paragraph between 12 points none

The following example toggles the space before formatting of the first paragraph in the selection. The macro retrieves the current space before value, and if the value is 12 points the space before formatting is removed (the [SpaceBefore](#) property is set to zero). If the space before value is anything other than 12, then **SpaceBefore** property is set to 12 points.

```
Sub ToggleParagraphSpace()  
  With Selection.Paragraphs(1)  
    If .SpaceBefore <> 0 Then  
      .SpaceBefore = 0  
    Else  
      .SpaceBefore = 6  
    End If  
  End With  
End Sub
```

Toggling bold formatting

The following example toggles bold formatting of the selected text.

```
Sub ToggleBold()  
    Selection.Font.Bold = wdToggle  
End Sub
```

Increasing the left margin by 0.5 inch

The following example increases the left and right margins by 0.5 inch. The **PageSetup** object contains all the page setup attributes of a document (left margin, bottom margin, paper size, and so on) as properties. The [LeftMargin](#) property is used to return and set the left margin setting. The [RightMargin](#) property is used to return and set the right margin setting.

```
Sub FormatMargins()  
    With ActiveDocument.PageSetup  
        .LeftMargin = .LeftMargin + InchesToPoints(0.5)  
        .RightMargin = .RightMargin + InchesToPoints(0.5)  
    End With  
End Sub
```



Editing Text

This topic includes Visual Basic examples related to the following tasks:

- [Determining whether text is selected](#)
- [Collapsing a selection or range](#)
- [Extending a selection or range](#)
- [Redefining a Range object](#)
- [Changing text](#)

For information about and examples of other editing tasks, see the following topics:

[Returning text from a document](#)

[Selecting text in a document](#)

[Inserting text in a document](#)

[Manipulating a portion of a document](#)

Determining whether text is selected

The [Type](#) property of the [Selection](#) object returns information about the type of selection. The following example displays a message if the selection is an insertion point.

```
Sub IsTextSelected()  
    If Selection.Type = wdSelectionIP Then MsgBox "Nothing is select  
End Sub
```

Collapsing a selection or range

Use the [Collapse](#) method to collapse a **Selection** or [Range](#) object to its beginning or ending point. The following example collapses the selection to an insertion point at the beginning of the selection.

```
Sub CollapseToBeginning()  
    Selection.Collapse Direction:=wdCollapseStart  
End Sub
```

The following example collapses the range to its ending point (after the first word) and adds new text.

```
Sub CollapseToEnd()  
    Dim rngWords As Range  
  
    Set rngWords = ActiveDocument.Words(1)  
    With rngWords  
        .Collapse Direction:=wdCollapseEnd  
        .Text = "(This is a test.) "  
    End With  
End Sub
```

Extending a selection or range

The following example uses the [MoveEnd](#) method to extend the end of the selection to include three additional words. The [MoveLeft](#), [MoveRight](#), [MoveUp](#), and [MoveDown](#) methods can also be used to extend a **Selection** object.

```
Sub ExtendSelection()  
    Selection.MoveEnd Unit:=wdWord, Count:=3  
End Sub
```

The following example uses the **MoveEnd** method to extend the range to include the first three paragraphs in the active document.

```
Sub ExtendRange()  
    Dim rngParagraphs As Range  
  
    Set rngParagraphs = ActiveDocument.Paragraphs(1).Range  
    rngParagraphs.MoveEnd Unit:=wdParagraph, Count:=2  
End Sub
```

Redefining a Range object

Use the [SetRange](#) method to redefine an existing **Range** object. For more information, see [Working with Range objects](#).

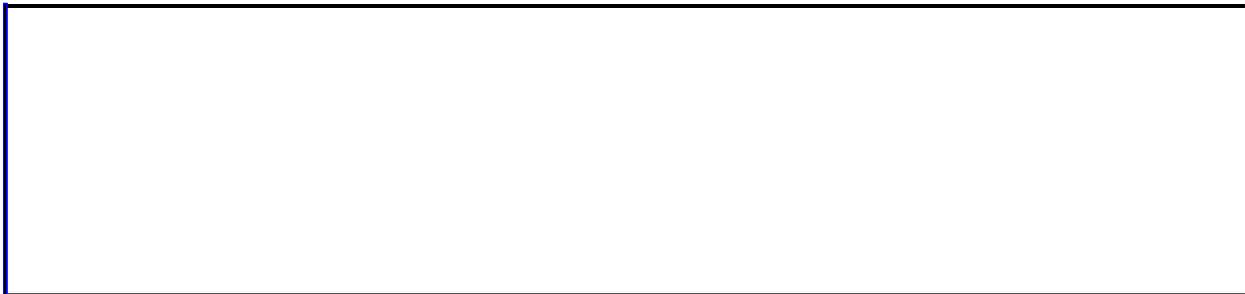
Changing text

You can change existing text by changing the contents of a range. The following instruction changes the first word in the active document by setting the [Text](#) property to "The."

```
Sub ChangeText()  
    ActiveDocument.Words(1).Text = "The "  
End Sub
```

You can also use the [Delete](#) method to delete existing text and then insert new text using the [InsertAfter](#) or [InsertBefore](#) method. The following example deletes the first paragraph in the active document and inserts new text.

```
Sub DeleteText()  
    Dim rngFirstParagraph As Range  
  
    Set rngFirstParagraph = ActiveDocument.Paragraphs(1).Range  
    With rngFirstParagraph  
        .Delete  
        .InsertAfter Text:="New text"  
        .InsertParagraphAfter  
    End With  
End Sub
```



Finding and Replacing Text or Formatting

Finding and replacing is exposed by the [Find](#) and [Replacement](#) objects. The **Find** object is available from the [Selection](#) and [Range](#) object. The find action differs slightly depending upon whether you access the **Find** object from the **Selection** or **Range** object.

Finding text and selecting it

If the **Find** object is accessed from the **Selection** object, the selection is changed when the find criteria is found. The following example selects the next occurrence of the word "Hello." If the end of the document is reached before the word "Hello" is found, the search is stopped.

```
With Selection.Find
    .Forward = True
    .Wrap = wdFindStop
    .Text = "Hello"
    .Execute
End With
```

The **Find** object includes properties that relate to the options in the **Find and Replace** dialog box (choose **Find** from the **Edit** menu). You can set the individual properties of the **Find** object or use arguments with the [Execute](#) method as shown in the following example.

```
Selection.Find.Execute FindText:="Hello", _
    Forward:=True, Wrap:=wdFindStop
```

Finding text without changing the selection

If the **Find** object is accessed from a **Range** object, the selection is not changed but the **Range** is redefined when the find criteria is found. The following example locates the first occurrence of the word "blue" in the active document. If the find operation is successful, the range is redefined and bold formatting is applied to the word "blue."

```
With ActiveDocument.Content.Find
    .Text = "blue"
    .Forward = True
    .Execute
    If .Found = True Then .Parent.Bold = True
End With
```

The following example performs the same result as the previous example using arguments of the **Execute** method.

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="blue", Forward:=True
If myRange.Find.Found = True Then myRange.Bold = True
```

Using the Replacement object

The **Replacement** object represents the replace criteria for a find and replace operation. The properties and methods of the **Replacement** object correspond to the options in the **Find and Replace** dialog box (**Edit** menu).

The **Replacement** object is available from the **Find** object. The following example replaces all occurrences of the word "hi" with "hello." The selection changes when the find criteria is found because the **Find** object is accessed from the **Selection** object.

```
With Selection.Find
    .ClearFormatting
    .Text = "hi"
    .Replacement.ClearFormatting
    .Replacement.Text = "hello"
    .Execute Replace:=wdReplaceAll, Forward:=True, _
        Wrap:=wdFindContinue
End With
```

The following example removes bold formatting in the active document. The **Bold** property is **True** for the **Find** object and **False** for the **Replacement** object. In order to find and replace formatting, set the find and replace text to empty strings ("") and set the **Format** argument of the **Execute** method to **True**. The selection remains unchanged because the **Find** object is accessed from a **Range** object (the **Content** property returns a **Range** object).

```
With ActiveDocument.Content.Find
    .ClearFormatting
    .Font.Bold = True
    With .Replacement
        .ClearFormatting
        .Font.Bold = False
    End With
    .Execute FindText:="", ReplaceWith:"", _
        Format:=True, Replace:=wdReplaceAll
End With
```



Miscellaneous Tasks

This topic includes Visual Basic examples for the following tasks:

- [Changing the view](#)
- [Setting text in a header or footer](#)
- [Setting options](#)
- [Changing the document layout](#)
- [Looping through paragraphs in a document](#)
- [Customizing menus and toolbars](#)

Changing the view

The [View](#) object includes properties and methods related to view attributes (show all, field shading, table gridlines, and so on) for a window or pane. The following example changes the view to print view.

```
Sub ChangeView()  
    ActiveDocument.ActiveWindow.View.Type = wdPrintView  
End Sub
```

Setting text in a header or footer

The [HeaderFooter](#) object is returned by the **Headers**, **Footers** and **HeaderFooter** properties. The following example changes the text of current page header.

```
Sub AddHeaderText()  
    With ActiveDocument.ActiveWindow.View  
        .SeekView = wdSeekCurrentPageHeader  
        Selection.HeaderFooter.Range.Text = "Header text"  
        .SeekView = wdSeekMainDocument  
    End With  
End Sub
```

This example creates a **Range** object (rngFooter) that references the primary footer for the first section in the active document. After the **Range** object is set, the existing footer text is deleted. The FILENAME field is added to the footer along with two tabs and the AUTHOR field.

```
Sub AddFooterText()  
    Dim rngFooter As Range  
    Set rngFooter = ActiveDocument.Sections(1) _  
        .Footers(wdHeaderFooterPrimary).Range  
    With rngFooter  
        .Delete  
        .Fields.Add Range:=rngFooter, Type:=wdFieldFileName, Text:="  
        .InsertAfter Text:=vbTab & vbTab  
        .Collapse Direction:=wdCollapseStart  
        .Fields.Add Range:=rngFooter, Type:=wdFieldAuthor  
    End With  
End Sub
```

Setting options

The [Options](#) object includes properties that correspond to items in the **Options** dialog box (**Tools** menu). The following example sets three application options for Word.

```
Sub SetOptions()  
  With Options  
    .AllowDragAndDrop = True  
    .ConfirmConversions = False  
    .MeasurementUnit = wdPoints  
  End With  
End Sub
```

Changing the document layout

The [PageSetup](#) contains all the page setup attributes of a document (left margin, bottom margin, paper size, and so on) as properties. The following example sets the margin values for the active document.

```
Sub ChangeDocumentLayout()  
  With ActiveDocument.PageSetup  
    .LeftMargin = InchesToPoints(0.75)  
    .RightMargin = InchesToPoints(0.75)  
    .TopMargin = InchesToPoints(1.5)  
    .BottomMargin = InchesToPoints(1)  
  End With  
End Sub
```

Looping through paragraphs in a document

This example loops through all of the paragraphs in the active document. If the space before setting for a paragraph is 6 points, this example changes the spacing to 12 points.

```
Sub LoopParagraphs()  
    Dim parCount As Paragraph  
    For Each parCount In ActiveDocument.Paragraphs  
        If parCount.SpaceBefore = 6 Then parCount.SpaceBefore = 12  
    Next parCount  
End Sub
```

For more information, see [Looping through a collection](#).

Customizing menus and toolbars

The [CommandBar](#) object represents both menus and toolbars. Use the [CommandBars](#) property with a menu or toolbar name to return a single [CommandBar](#) object. The [Controls](#) property returns a [CommandBarControls](#) object that refers to the items on the specified command bar. The following example adds the Word Count command to the **Standard** menu.

```
Sub AddToolbarItem()  
    Dim btnNew As CommandBarButton  
    CustomizationContext = NormalTemplate  
    Set btnNew = CommandBars("Standard").Controls.Add _  
        (Type:=msoControlButton, ID:=792, Before:=6)  
    With btnNew  
        .BeginGroup = True  
        .FaceId = 700  
        .TooltipText = "Word Count"  
    End With  
End Sub
```

The following example adds the **Double Underline** command to the **Formatting** toolbar.

```
Sub AddDoubleUnderlineButton()  
    CustomizationContext = NormalTemplate  
    CommandBars("Formatting").Controls.Add _  
        Type:=msoControlButton, ID:=60, Before:=7  
End Sub
```

Turn on the macro recorder and customize a menu or toolbar to determine the **ID** value for a particular command (for example, ID 60 is the **Double Underline** command).



Working with Tables

This topic includes Visual Basic examples related to the following tasks:

- [Creating a table, inserting text, and applying formatting](#)
- [Inserting text into a table cell](#)
- [Returning text from a table cell without returning the end of cell marker](#)
- [Converting text to a table](#)
- [Returning the contents of each table cell](#)
- [Copying all tables in the active document into a new document](#)

Creating a table, inserting text, and applying formatting

The following example inserts a four column, three row table at the beginning of the active document. The **For Each...Next** structure is used to step through each cell in the table. Within the **For Each...Next** structure, the **InsertAfter** method is used to add text to the table cells (Cell 1, Cell 2, and so on).

```
Sub CreateNewTable()  
    Dim docActive As Document  
    Dim tblNew As Table  
    Dim celTable As Cell  
    Dim intCount As Integer  
  
    Set docActive = ActiveDocument  
    Set tblNew = docActive.Tables.Add( _  
        Range:=docActive.Range(Start:=0, End:=0), NumRows:=3, _  
        NumColumns:=4)  
    intCount = 1  
  
    For Each celTable In tblNew.Range.Cells  
        celTable.Range.InsertAfter "Cell " & intCount  
        intCount = intCount + 1  
    Next celTable  
  
    tblNew.AutoFormat Format:=wdTableFormatColorful2, _  
        ApplyBorders:=True, ApplyFont:=True, ApplyColor:=True  
End Sub
```

Inserting text into a table cell

The following example inserts text into the first cell of the first table in the active document. The [Cell](#) method returns a single **Cell** object. The [Range](#) property returns a **Range** object. The [Delete](#) method is used to delete the existing text and the [InsertAfter](#) method inserts the "Cell 1,1" text.

```
Sub InsertTextInCell()  
    If ActiveDocument.Tables.Count >= 1 Then  
        With ActiveDocument.Tables(1).Cell(Row:=1, Column:=1).Range  
            .Delete  
            .InsertAfter Text:="Cell 1,1"  
        End With  
    End If  
End Sub
```

Returning text from a table cell without returning the end of cell marker

The following examples return and display the contents of each cell in the first row of the first document table.

```
Sub ReturnTableText()  
    Dim tblOne As Table  
    Dim celTable As Cell  
    Dim rngTable As Range  
  
    Set tblOne = ActiveDocument.Tables(1)  
    For Each celTable In tblOne.Rows(1).Cells  
        Set rngTable = ActiveDocument.Range(Start:=celTable.Range.Start  
            End:=celTable.Range.End - 1)  
        MsgBox rngTable.Text  
    Next celTable  
End Sub
```

```
Sub ReturnCellText()  
    Dim tblOne As Table  
    Dim celTable As Cell  
    Dim rngTable As Range  
  
    Set tblOne = ActiveDocument.Tables(1)  
    For Each celTable In tblOne.Rows(1).Cells  
        Set rngTable = celTable.Range  
        rngTable.MoveEnd Unit:=wdCharacter, Count:=-1  
        MsgBox rngTable.Text  
    Next celTable  
End Sub
```

Converting existing text to a table

The following example inserts tab-delimited text at the beginning of the active document and then converts the text to a table.

```
Sub ConvertExistingText()  
  With Documents.Add.Content  
    .InsertBefore "one" & vbTab & "two" & vbTab & "three" & vbCr  
    .ConvertToTable Separator:=Chr(9), NumRows:=1, NumColumns:=3  
  End With  
End Sub
```

Returning the contents of each table cell

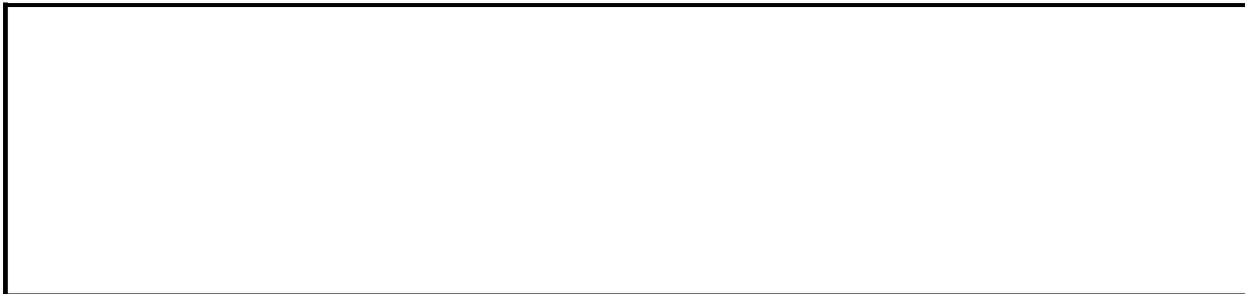
The following example defines an array equal to the number of cells in the first document table (assuming **Option Base 1**). The **For Each...Next** structure is used to return the contents of each table cell and assign the text to the corresponding array element.

```
Sub ReturnCellContentsToArray()  
    Dim intCells As Integer  
    Dim celTable As Cell  
    Dim strCells() As String  
    Dim intCount As Integer  
    Dim rngText As Range  
  
    If ActiveDocument.Tables.Count >= 1 Then  
        With ActiveDocument.Tables(1).Range  
            intCells = .Cells.Count  
            ReDim strCells(intCells)  
            intCount = 1  
            For Each celTable In .Cells  
                Set rngText = celTable.Range  
                rngText.MoveEnd Unit:=wdCharacter, Count:=-1  
                strCells(intCount) = rngText  
                intCount = intCount + 1  
            Next celTable  
        End With  
    End If  
End Sub
```

Copying all tables in the active document into a new document

This example copies the tables from the current document into a new document.

```
Sub CopyTablesToNewDoc()  
    Dim docOld As Document  
    Dim rngDoc As Range  
    Dim tblDoc As Table  
  
    If ActiveDocument.Tables.Count >= 1 Then  
        Set docOld = ActiveDocument  
        Set rngDoc = Documents.Add.Range(Start:=0, End:=0)  
        For Each tblDoc In docOld.Tables  
            tblDoc.Range.Copy  
            With rngDoc  
                .Paste  
                .Collapse Direction:=wdCollapseEnd  
                .InsertParagraphAfter  
                .Collapse Direction:=wdCollapseEnd  
            End With  
        Next  
    End If  
End Sub
```



Predefined Bookmarks

Microsoft Word sets and automatically updates a number of reserved bookmarks. You can use these predefined bookmarks just as you use the ones that you place in documents, except that you don't have to set them and they are not listed on the **Go To** tab in the **Find and Replace** dialog box (**Edit** menu).

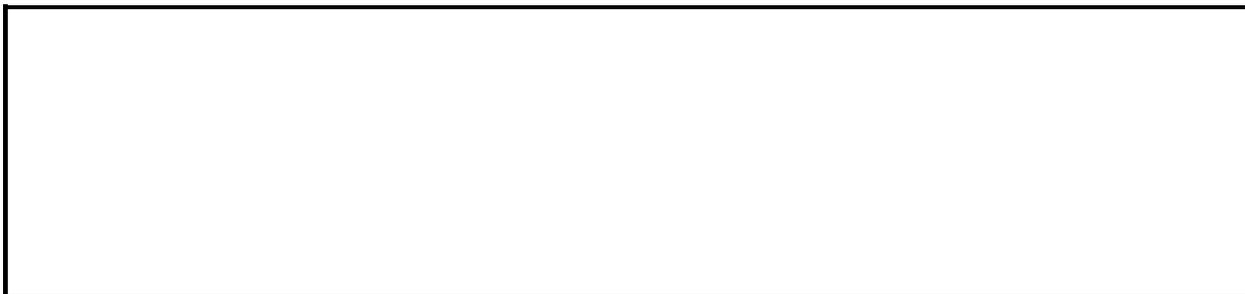
You can use predefined bookmarks with the [Bookmarks](#) property. The following example sets the bookmark named "currpara" to the location marked by the predefined bookmark named "\Para."

```
ActiveDocument.Bookmarks("\Para").Copy "currpara"
```

The following table describes the predefined bookmarks available in Word.

Bookmark	Description
\Sel	Current selection or the insertion point.
\PrevSel1	Most recent selection where editing occurred; going to this bookmark is equivalent to running the GoBack method once.
\PrevSel2	Second most recent selection where editing occurred; going to this bookmark is equivalent to running the GoBack method twice.
\StartOfSel	Start of the current selection.
\EndOfSel	End of the current selection.
\Line	Current line or the first line of the current selection. If the insertion point is at the end of a line that is not the last line in the paragraph, the bookmark includes the entire next line.
\Char	Current character, which is the character following the insertion point if there is no selection, or the first character of the selection.
\Para	Current paragraph, which is the paragraph containing the insertion point or, if more than one paragraph is selected, the first paragraph of the selection. Note that if the insertion point or selection is in the last paragraph of the document, the

	"\Para" bookmark does not include the paragraph mark.
\Section	Current section, including the break at the end of the section, if any. The current section contains the insertion point or selection. If the selection contains more than one section, the "\Section" bookmark is the first section in the selection.
\Doc	Entire contents of the active document, with the exception of the final paragraph mark.
\Page	Current page, including the break at the end of the page, if any. The current page contains the insertion point. If the current selection contains more than one page, the "\Page" bookmark is the first page of the selection. Note that if the insertion point or selection is in the last page of the document, the "\Page" bookmark does not include the final paragraph mark.
\StartOfDoc	Beginning of the document.
\EndOfDoc	End of the document.
\Cell	Current cell in a table, which is the cell containing the insertion point. If one or more cells of a table are included in the current selection, the "\Cell" bookmark is the first cell in the selection.
\Table	Current table, which is the table containing the insertion point or selection. If the selection includes more than one table, the "\Table" bookmark is the entire first table of the selection, even if the entire table is not selected.
\HeadingLevel	The heading that contains the insertion point or selection, plus any subordinate headings and text. If the current selection is body text, the "\HeadingLevel" bookmark includes the preceding heading, plus any headings and text subordinate to that heading.



Creating Frames Pages

In Microsoft Word, you can use frames in your Web page design to make your information organized and easy to access. A frames page, also called a frameset, is a Web page that is divided into two or more frames, each of which points to another Web page. A frame on a frames page can also point to another frames page. For information about creating frames and frames pages in the Word user interface, see [Create frames and frames pages](#).

Frames and frames pages are created with a series of HTML tags. The Visual Basic object model for working with frames and frames pages is best understood by examining their HTML tags.

Frames pages in HTML

In HTML, frames pages and the frames they contain are built using a hierarchical set of <FRAMESET> and <FRAME> tags. A frameset can contain both frames and other framesets. For example, the following HTML creates a frameset with a frame on top and a frameset immediately below it. That frameset contains a frame on the left and a frameset on the right. That frameset contains two frames, one on top of the other.

```
<FRAMESET ROWS="100, *">
  <FRAME NAME=top SRC="banner.htm">
  <FRAMESET COLS="20%, *">
    <FRAME NAME=left SRC="contents.htm">
    <FRAMESET ROWS="75%, *">
      <FRAME NAME=main SRC="main.htm">
      <FRAME NAME=bottom SRC="footer.htm">
    </FRAMESET>
  </FRAMESET>
</FRAMESET>
```

Note To better understand the preceding HTML example, paste the example into a blank text document, rename the document "framespage.htm", and open the document in Word or in a Web browser.

The Frameset Object

The [Frameset](#) object encompasses the functionality of both tags. Each **Frameset** object is either of type **wdFramesetTypeFrameset** or **wdFramesetTypeFrame**, which represent the HTML tags <FRAMESET> and <FRAME> respectively. Properties beginning with "Frameset" apply to **Frameset** objects of type **wdFramesetTypeFrameset** ([FramesetBorderColor](#) and [FramesetBorderWidth](#)). Properties beginning with "Frame" apply to **Frameset** objects of type **wdFramesetTypeFrame** ([FrameDefaultURL](#), [FrameDisplayBorders](#), [FrameLinkToFile](#), [FrameName](#), [FrameResizable](#), and [FrameScrollbarType](#)).

Traversing the Frameset Object Hierarchy

Because frames pages are defined as a hierarchical set of HTML tags, the object model for accessing **Frameset** objects is also hierarchical. Use the [ChildFramesetItem](#) and [ParentFrameset](#) properties to traverse the hierarchy of **Frameset** objects. For example,

```
MyFrameset.ChildFramesetItem(n)
```

returns a **Frameset** object corresponding to the *n*th first-level <FRAMESET> or <FRAME> tag between the <FRAMESET> and </FRAMESET> tags corresponding to MyFrameset.

If MyFrameset is a **Frameset** object corresponding to the outermost <FRAMESET> tags in the preceding HTML example, MyFrameset.ChildFramesetItem(1) returns a **Frameset** object of type **wdFramesetTypeFrame** that corresponds to the frame named "top." Similarly, MyFrameset.ChildFramesetItem(2) returns a **Frameset** object of type **wdFramesetTypeFrameset**, itself containing two **Frameset** objects: the first object corresponds to the frame named "left," the second is of type **wdFramesetTypeFrameset**.

Frameset objects of type **wdFramesetTypeFrame** have no child frames, while objects of **wdFramesetTypeFrameset** have at least one.

The following Visual Basic example displays the names of the four frames defined in the preceding HTML example.

```
Dim MyFrameset As Frameset
Dim Name1 As String
Dim Name2 As String
Dim Name3 As String
Dim Name4 As String

Set MyFrameset = ActiveWindow.Document.Frameset

With MyFrameset
    Name1 = .ChildFramesetItem(1).FrameName
    With .ChildFramesetItem(2)
        Name2 = .ChildFramesetItem(1).FrameName
```

```
        With .ChildFramesetItem(2)
            Name3 = .ChildFramesetItem(1).FrameName
            Name4 = .ChildFramesetItem(2).FrameName
        End With
    End With
End With

Debug.Print Name1, Name2, Name3, Name4
```

Individual Frames and the Entire Frames Page

To return the **Frameset** object associated with a particular frame on a frames page, use the **Frameset** property of a **Pane** object. For example,

```
ActiveWindow.Panes(1).Frameset
```

returns the **Frameset** object that corresponds to the first frame of the frames page.

The frames page is itself a document separate from the documents that make up the content of the individual frames. The **Frameset** object associated with a frames page is accessed from its corresponding **Document** object, which in turn is accessed from the **Window** object in which the frames page appears. For example,

```
ActiveWindow.Document.Frameset
```

returns the **Frameset** object for the frames page in the current window.

Note When working with frames pages, the **ActiveDocument** property returns the **Document** object associated with the frame in the active pane, not the entire frames page.

Creating a Frames Page and Its Content from Scratch

This example creates a new frames page with three frames, adds text to each frame, and sets the background color for each frame. It inserts two hyperlinks into the Left frame: the first hyperlink opens a document called One.htm in the Main frame, and the second opens a document called Two.htm in the entire window. For these hyperlinks to work, you must create files called One.htm and Two.htm or change the strings to the names of existing files.

Note As each frame is created, Word creates a new document whose content will be loaded into the new frame. The example saves the frames page which automatically saves the documents associated with each of the three frames.

```
Sub FramesetExample1()  
  
    ' Create new frames page.  
    Documents.Add DocumentType:=wdNewFrameset  
  
    With ActiveWindow  
        ' Add text and color to first frame.  
        Selection.TypeText Text:="BANNER FRAME"  
        With ActiveDocument.Background.Fill  
            .ForeColor.RGB = RGB(204, 153, 255)  
            .Visible = msoTrue  
        End With  
  
        ' Add new frame below top frame.  
        .ActivePane.Frameset.AddNewFrame _  
            wdFramesetNewFrameBelow  
        ' Add text and color to bottom frame.  
        .ActivePane.Frameset.FrameName = "main"  
        Selection.TypeText Text:="MAIN FRAME"  
        With ActiveDocument.Background.Fill  
            .ForeColor.RGB = RGB(0, 128, 128)  
            .Visible = msoTrue  
        End With  
  
        ' Add new frame to left of bottom frame.  
        .ActivePane.Frameset.AddNewFrame _  
            wdFramesetNewFrameLeft  
        ' Set the width to 25% of the window width.  
        With .ActivePane.Frameset  
            .WidthType = wdFramesetSizeTypePercent
```

```

        .Width = 25
        .FrameName = "left"
    End With
    ' Add text and color to left frame.
    Selection.TypeText Text:="LEFT FRAME"
    With ActiveDocument.Background.Fill
        .ForeColor.RGB = RGB(204, 255, 255)
        .Visible = msoTrue
    End With
    Selection.TypeParagraph
    Selection.TypeParagraph
    ' Add hyperlinks to left frame.
    With ActiveDocument.Hyperlinks
        .Add Anchor:=Selection.Range, _
            Address:="one.htm", Target:="main"
        Selection.TypeParagraph
        Selection.TypeParagraph
        .Add Anchor:=Selection.Range, _
            Address:="two.htm", Target:="_top"
    End With

    ' Activate top frame.
    .Panes(1).Activate
    ' Set the height to 1 inch.
    With .ActivePane.Frameset
        .HeightType = wdFramesetSizeTypeFixed
        .Height = InchesToPoints(1)
        .FrameName = "top"
    End With

    ' Save the frames page and its associated files.
    .Document.SaveAs FileName:="default.htm", _
        FileFormat:=wdFormatHTML
End With

End Sub

```

Creating a Frames Page that Displays Content from Existing Files

This example creates a frames page similar to the one above, but sets the default URL for each frame to an existing document so that the content of that document is displayed in the frame. For this example to work, you must create files called Main.htm, Left.htm, and Banner.htm or change the strings in the example to the names of existing files.

```
Sub FramesetExample2()

    ' Create new frames page.
    Documents.Add DocumentType:=wdNewFrameset

    With ActiveWindow
        ' Add new frame below top frame.
        .ActivePane.Frameset.AddNewFrame _
            wdFramesetNewFrameBelow
        ' Set the name and initial page for the frame.
        With .ActivePane.Frameset
            .FrameName = "main"
            .FrameDefaultURL = "main.htm"
        End With

        ' Add new frame to left of bottom frame.
        .ActivePane.Frameset.AddNewFrame _
            wdFramesetNewFrameLeft
        With .ActivePane.Frameset
            ' Set the width to 25% of the window width.
            .WidthType = wdFramesetSizeTypePercent
            .Width = 25
            ' Set the name and initial page for the frame.
            .FrameName = "left"
            .FrameDefaultURL = "left.htm"
        End With

        ' Activate top frame.
        .Panes(1).Activate
        With .ActivePane.Frameset
            ' Set the height to 1 inch.
            .HeightType = wdFramesetSizeTypeFixed
            .Height = InchesToPoints(1)
            ' Set the name and initial page for the frame.
```

```
        .FrameName = "top"  
        .FrameDefaultURL = "banner.htm"  
    End With  
  
    ' Save the frameset.  
    .Document.SaveAs FileName:="default.htm", _  
        FileFormat:=wdFormatHTML  
    End With  
  
End Sub
```



Learn About Language-Specific Information

Language-specific Help topics apply only if the language-specific feature is available. Learn about [working in another language](#) or [installing the proofing tools for another language](#), or see your system administrator for more information.



Working with the Selection Object

When you work on a document in Word, you usually select text and then perform an action, such as formatting the text or typing text. In Visual Basic, it is usually not necessary to select text before modifying the text. Instead, you create a [Range](#) object that refers to a specific portion of the document. For information on defining **Range** objects, see [Working with Range objects](#). However, when you want your code to respond to or change the selection, you can do so with the [Selection](#) object.

The [Select](#) method activates an object. For example, the following instruction selects the first word in the active document.

```
Sub SelectFirstWord()  
    ActiveDocument.Words(1).Select  
End Sub
```

For more information, see [Selecting text in a document](#).

The **Selection** property returns a [Selection](#) object that represents the active selection in a document window pane. There can only be one **Selection** object per document window pane and only one **Selection** object can be active. For example, the following example changes the paragraph formatting of the paragraphs in the selection.

```
Sub FormatSelection()  
    Selection.Paragraphs.LeftIndent = InchesToPoints(0.5)  
End Sub
```

For example, the following example inserts the word "Hello" after the selection.

```
Sub InsertTextAfterSelection()  
    Selection.InsertAfter Text:="Hello "  
End Sub
```

The following example applies bold formatting to the selected text.

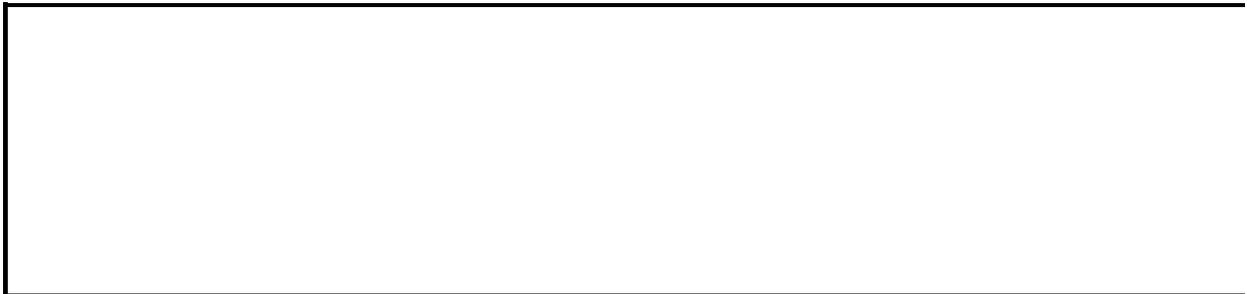
```
Sub BoldSelectedText()  
    Selection.Font.Bold = True  
End Sub
```

The macro recorder will often create a macro that uses the **Selection** property. The following example was created using the macro recorder. The macro applies bold formatting to the first two words in the document.

```
Sub Macro()  
    Selection.HomeKey Unit:=wdStory  
    Selection.MoveRight Unit:=wdWord, Count:=2, Extend:=wdExtend  
    Selection.Font.Bold = wdToggle  
End Sub
```

The following example accomplishes the same task without using the **Selection** property.

```
Sub WorkingWithRanges()  
    ActiveDocument.Range(Start:=0, _  
        End:=ActiveDocument.Words(2).End).Bold = True  
End Sub
```



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 a **ShapeRange** object.

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.

Note that 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 shapes in the selection in the active window. If the shapes haven't been previously grouped and ungrouped, this example will fail.

```
ActiveDocument.ActiveWindow.Selection.ShapeRange.Regroup
```



Using Events with the Document Object

The [Document](#) object supports three events: Close, New and Open. 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 Normal project or document project in the Project Explorer window, double-click **ThisDocument**. (In Folder view, **ThisDocument** is located in the **Microsoft Word 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.

The following example shows a [New](#) event procedure in the Normal project that will run when a new document based on the Normal template is created.

```
Private Sub Document_New()  
    MsgBox "New document was created"  
End Sub
```

The following example shows a [Close](#) event procedure in a document project that runs only when that document is closed.

```
Private Sub Document_Close()  
    MsgBox "Closing the document"  
End Sub
```

Unlike [auto macros](#), event procedures in the Normal template don't have a global scope. For example, event procedures in the Normal template only occur if the attached template is the Normal template.

If an auto macro exists in a document and the attached template, only the auto macro stored in the document will execute. If an event procedure for a document event exists in a document and its attached template, both event procedures will run.

Remarks

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 Word.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_DocumentChange()
```

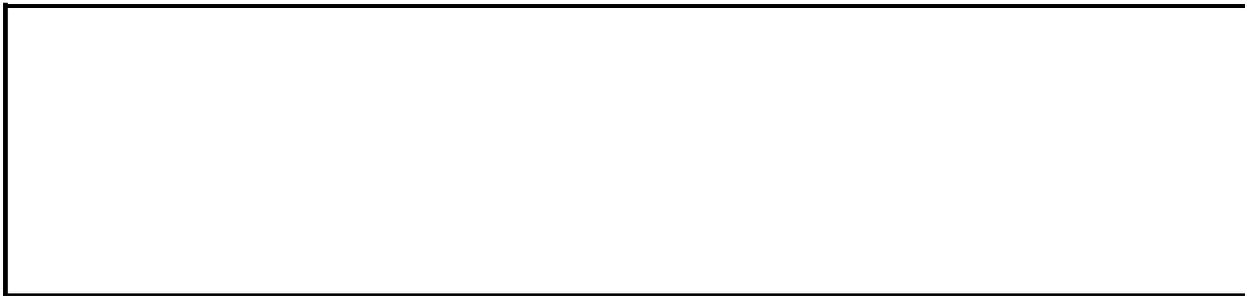
```
End Sub
```

Initialize the Declared Object

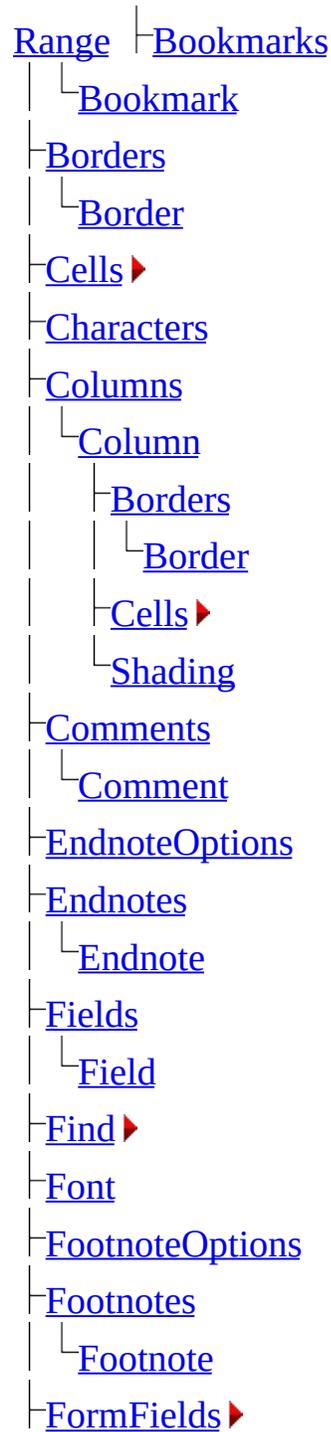
Before the procedure will run, you must connect the declared object in the class module (App in this example) with the **Application** object. You can do this with the following code from any module.

```
Dim X As New EventClassModule
Sub Register_Event_Handler()
    Set X.App = Word.Application
End Sub
```

Run the Register_Event_Handler procedure. After the procedure is run, the App object in the class module points to the Microsoft Word **Application** object, and the event procedures in the class module will run when the events occur.



Microsoft Word Objects (Range)



- [Frames](#)
 - [Frame](#)
- [HTMLDivisions](#)
 - [HTMLDivision](#)
 - [Borders](#)
- [Hyperlinks](#)
 - [Hyperlink](#)
- [InlineShapes](#) ▶
- [ListFormat](#)
 - [InlineShape](#) ▶
 - [List](#)
 - [ListTemplate](#)
- [ListParagraphs](#)
 - [Paragraph](#) ▶
- [PageSetup](#)
 - [LineNumbering](#)
 - [TextColumns](#)
 - [TextColumn](#)
- [ParagraphFormat](#)
- [Paragraphs](#) ▶
- [ReadabilityStatistics](#)
 - [ReadabilityStatistic](#)
- [Revisions](#)
 - [Revision](#)
- [Rows](#)
 - [Row](#)
- [Scripts](#)
 - [Script](#)
- [Sections](#) ▶
- [Sentences](#)

- | [Shading](#)
- | [ShapeRange \(Shape\)](#) ▶
- | [ProofreadingErrors](#)
- | [Style](#)
- | [Subdocuments](#)
 - | [Subdocument](#)
- | [SynonymInfo](#)
- | [Tables](#) ▶
- | [TextRetrievalMode](#)
- | [Words](#)

Legend

Object and collection

Object only

▶ Click red arrow to expand chart



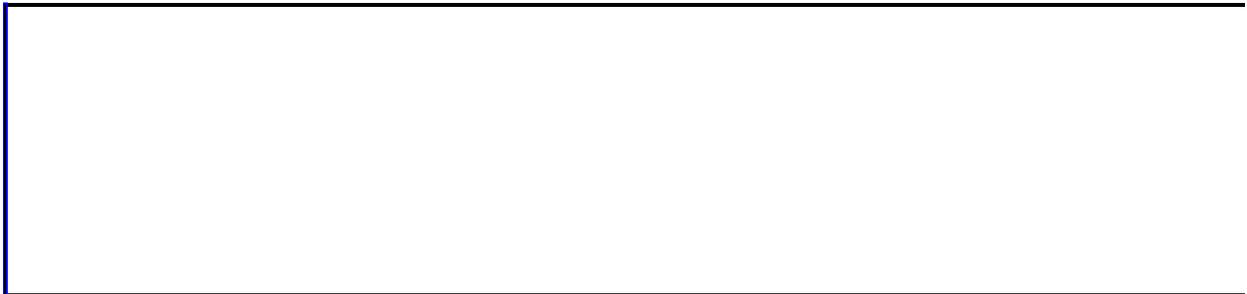
Microsoft Word Objects (FormFields)



Legend

Object and collection

Object only



Microsoft Word Objects (InlineShapes)



Legend

Object and collection

Object only



Microsoft Word Objects (Paragraphs)



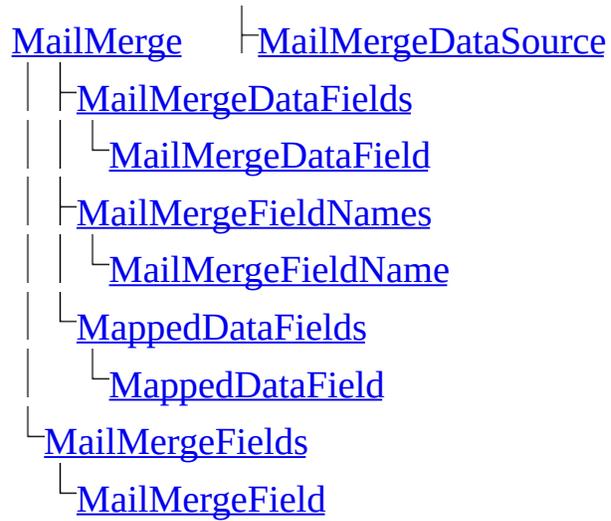
Legend

Object and collection

Object only



Microsoft Word Objects (MailMerge)



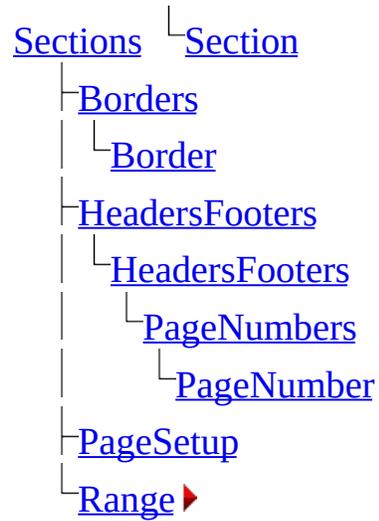
Legend

Object and collection

Object only



Microsoft Word Objects (Section)

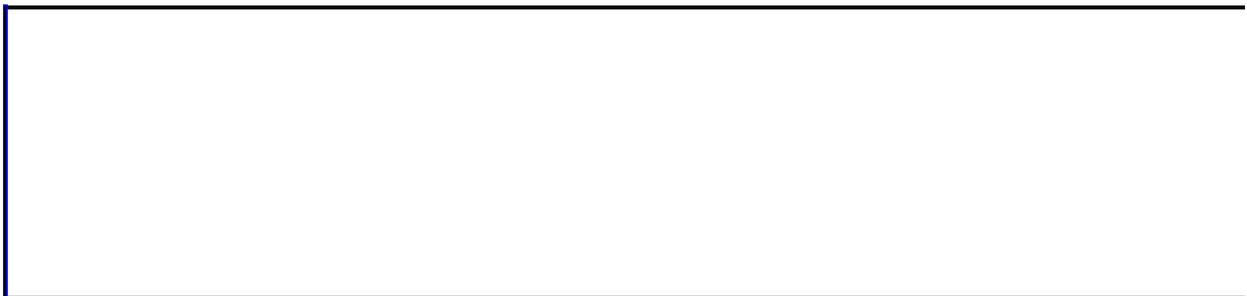


Legend

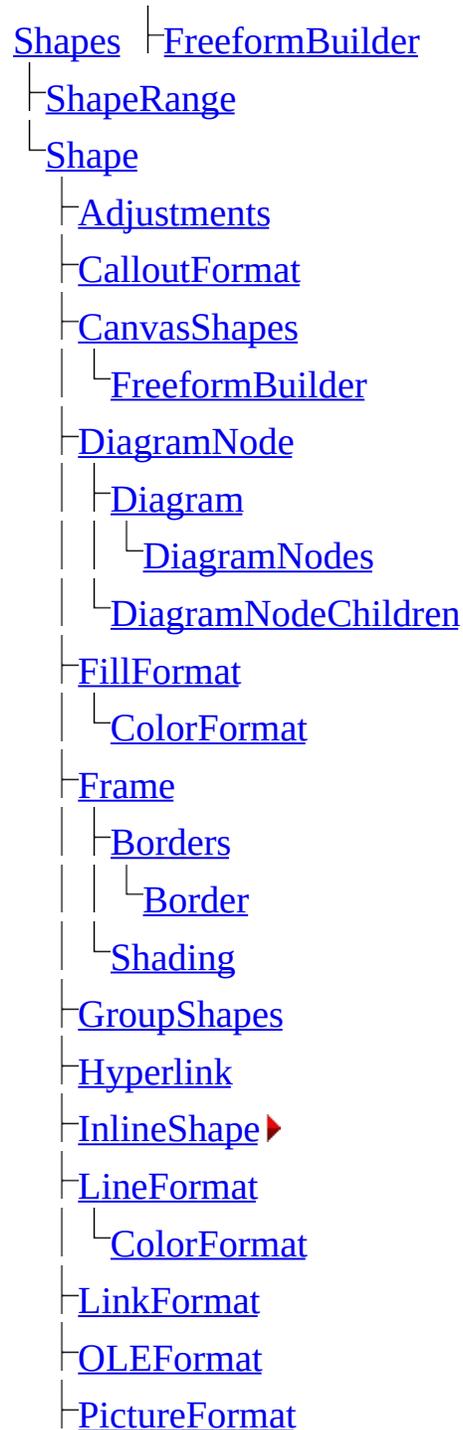
Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (Shapes)



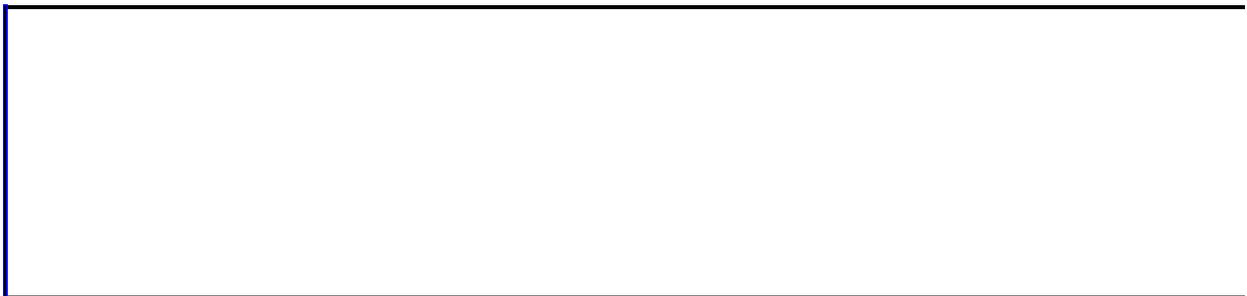
- | [Script](#)
- | [ShadowFormat](#)
 - | [ColorFormat](#)
- | [ShapeNodes](#)
 - | [ShapeNode](#)
- | [ShapeRange](#)
- | [TextEffectFormat](#)
- | [TextFrame](#)
- | [ThreeDFormat](#)
 - | [ColorFormat](#)
- | [WrapFormat](#)

Legend

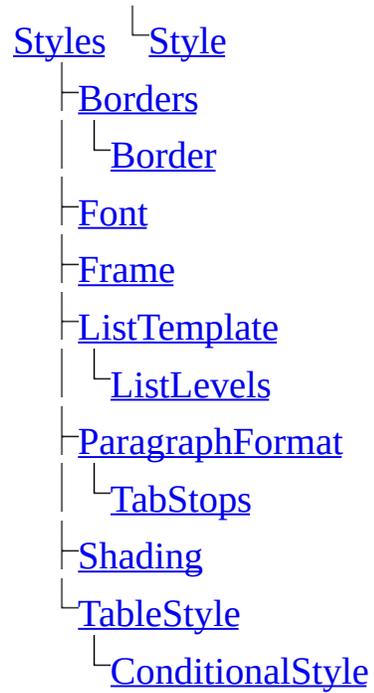
Object and collection

Object only

▶ Click red arrow to expand chart



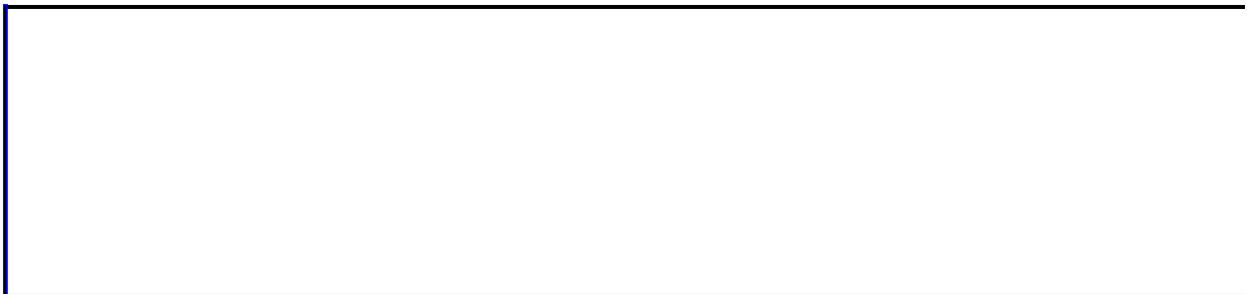
Microsoft Word Objects (Styles)



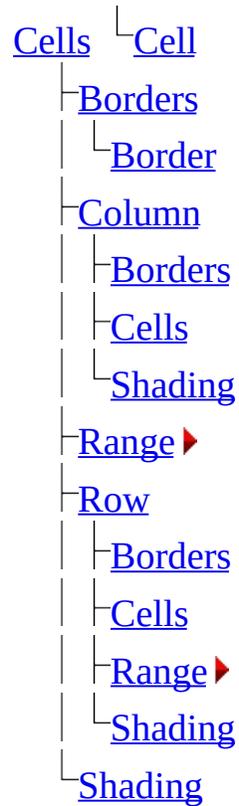
Legend

Object and collection

Object only



Microsoft Word Objects (Cells)



Legend

Object and collection

Object only

▶ Click red arrow to expand chart



Microsoft Word Objects (Find)

[Find](#) | [Font](#)
| [Frame](#)
| [ParagraphFormat](#)
| [Replacement](#)

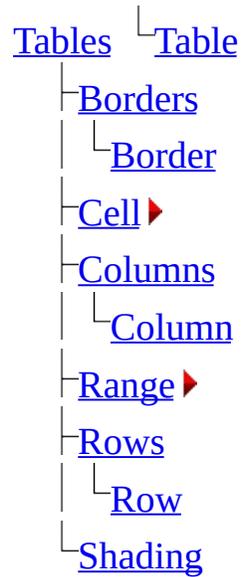
Legend

Object and collection

Object only



Microsoft Word Objects (Tables)

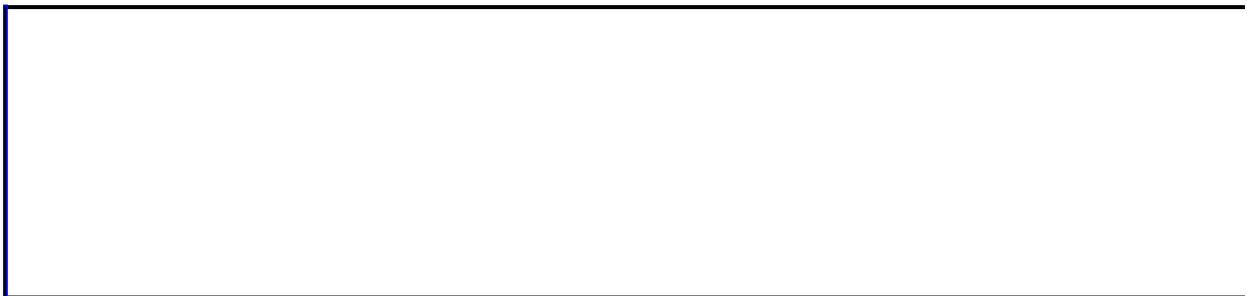


Legend

Object and collection

Object only

▶ Click red arrow to expand chart



XML Property

Returns a **String** that represents the related XML for a smart tag. Read-only.

expression.XML

expression Required. An expression that returns a [SmartTag](#) object.

Example

This example displays the XML information for the first smart tag in the active document. This example assumes that the active document contains at least one smart tag.

```
Sub SmartTagXml()  
    MsgBox "The XML information for this smart tag is : " & _  
        ActiveDocument.SmartTags(1).XML  
End Sub
```



CheckIn Method

Returns a document from a local computer to a server, and sets the local document to read-only so that it cannot be edited locally.

expression.**CheckIn**(*SaveChanges*, *MakePublic*, *Comments*)

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

SaveChanges Optional **Boolean**. **True** saves the document to the server location. The default is **True**.

MakePublic Optional **Boolean**. **True** allows the user to perform a publish on the document after being checked in. This submits the document for the approval process, which can eventually result in a version of the document being published to users with read-only rights to the document (only applies if **SaveChanges** equals **True**).

Comments Optional **Variant**. Comments for the revision of the document being checked in (only applies if **SaveChanges** equals **True**).

Remarks

To take advantage of the collaboration features built into Word, documents must be stored on a Microsoft SharePoint Portal Server.

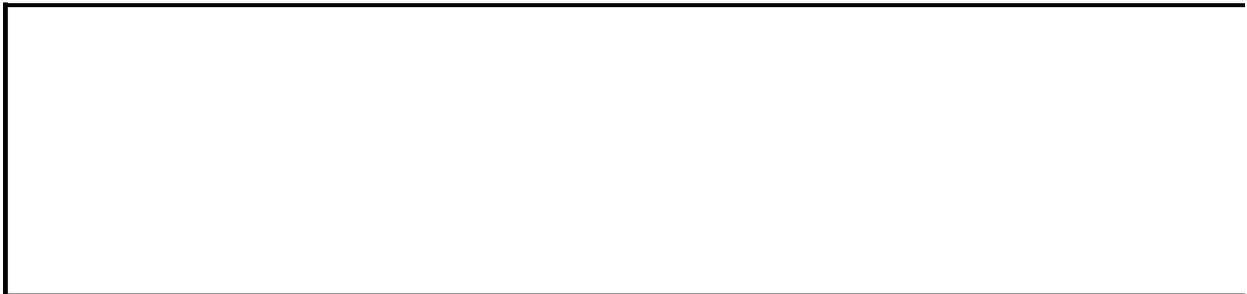
Example

This example checks the server to see if the specified document can be checked in. If it can be, it saves and closes the document and checks it back into the server.

```
Sub CheckInOut(docCheckIn As String)
    If Documents(docCheckIn).CanCheckin = True Then
        Documents(docCheckIn).CheckIn
        MsgBox docCheckIn & " has been checked in."
    Else
        MsgBox "This file cannot be checked in " &
            "at this time. Please try again later."
    End If
End Sub
```

To call the CheckInOut subroutine, use the following subroutine and replace the "http://servername/workspace/report.doc" file name with an actual file located on a server mentioned in the Remarks section above.

```
Sub CheckDocInOut()
    Call CheckInOut (docCheckIn:="http://servername/workspace/report
End Sub
```



↳ [Show All](#)

BaseLineAlignment Property

Returns or sets a [WdBaselineAlignment](#) constant that represents the vertical position of fonts on a line. Read/write.

WdBaselineAlignment can be one of these WdBaselineAlignment constants.

wdBaselineAlignAuto

wdBaselineAlignCenter

wdBaselineAlignTop

wdBaselineAlignBaseline

wdBaselineAlignFarEast50

expression.**BaseLineAlignment**

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

Example

This example sets Microsoft Word to automatically adjust the baseline font alignment in the active document.

```
ActiveDocument.BaseLineAlignment = wdBaselineAlignAuto
```



Visual Basic Equivalents A

A B C D E F G H I J K L M N O P R S T U V W Y

A

<i>Abs(number)</i>	<i>Abs(number)</i> <i>Windows(name).</i> Activate
<i>Activate name</i>	' <i>or</i> <i>Documents(name).</i> Activate <i>ActiveDocument.Shapes(1).OLEFormat.</i> Activate
<i>ActivateObject</i>	' <i>or</i> <i>ActiveDocument.InlineShapes(1).OLEFormat.</i> Activat
<i>AddAddIn</i>	<i>Addins.</i> Add
<i>AddAddress</i>	<i>Application.</i> AddAddress
<i>AddButton</i>	<i>CommandBars(name).Controls.</i> Add
<i>AddDropDownItem</i>	<i>ActiveDocument.FormFields(1).DropDown.ListEntry</i> <i>Addins(name).</i> Installed = True
<i>AddInState</i>	<i>state = Addins(name).</i> Installed
<i>state = AddInState(name)</i>	<i>state = Addins(name).</i> Compiled <i>state = Addins(name).</i> AutoLoad
<i>AllCaps, AllCaps()</i>	<i>Selection.Font.</i> AllCaps = True <i>x = Selection.Font.</i> AllCaps

Next aTask

AppHide <i>name</i>	Tasks(<i>name</i>). Visible = False
AppInfo\$(1)	MsgBox System. OperatingSystem & Chr(32) & System
AppInfo\$(2)	x = Application. Version
AppInfo\$(3)	x = Application. SpecialMode
AppInfo\$(4)	x = Application. Left
AppInfo\$(5)	x = Application. Top
AppInfo\$(6)	x = Application. UsableWidth
AppInfo\$(7)	x = Application. UsableHeight
AppInfo\$(8)	x = Application. WindowState (wdWindowStateMaxim
AppInfo\$(9)	x = WordBasic .[AppInfo\$(9)]
AppInfo\$(10)	x = WordBasic .[AppInfo\$(10)]
AppInfo\$(13)	x = System. MathCoprocessorInstalled
AppInfo\$(14)	x = Application. MouseAvailable
AppInfo\$(15)	x = System. FreeDiskSpace
AppInfo\$(16)	x = Application. International (wdProductLanguageID)
AppInfo\$(17)	x = Application. International (wdListSeparator)
AppInfo\$(18)	x = Application. International (wdDecimalSeparator)
AppInfo\$(19)	x = Application. International (wdThousandsSeparator)
AppInfo\$(20)	x = Application. International (CurrencyCode)
AppInfo\$(21)	x = Application. International (wd24HourClock)

AppInfo\$(22)	x = Application. International (wdInternationalAM)
AppInfo\$(23)	x = Application. International (wdInternationalPM)
AppInfo\$(24)	x = Application. International (wdTimeSeparator)
AppInfo\$(25)	x = Application. International (wdDateSeparator)
AppInfo\$(26)	x = WordBasic . [AppInfo\$](26)
AppInfo\$(27)	x = Application. LanguageSettings.LanguageID (msoLang)
AppIsRunning(<i>name</i>)	Tasks(<i>name</i>). Exists
AppMaximize <i>name</i>	Tasks(<i>name</i>). WindowState = wdWindowStateMaximi
AppMaximize	Application. WindowState = wdWindowStateMaximiz
AppMinimize <i>name</i>	Tasks(<i>name</i>). WindowState = wdWindowStateMinimiz
AppMinimize	Application. WindowState = wdWindowStateMinimiz
AppMove <i>name</i> , <i>horizpos</i> , <i>vertpos</i>	Tasks(<i>name</i>). Move Left:= <i>horizpos</i> , Top:= <i>vertpos</i>
AppMove <i>horizpos</i> , <i>vertpos</i>	Application. Move Left:= <i>horizpos</i> , Top:= <i>vertpos</i>
AppRestore <i>name</i>	Tasks(<i>name</i>). WindowState = wdWindowStateNormal
AppRestore	Application. WindowState = wdWindowStateNormal
AppSendMessage	Tasks(<i>name</i>). SendWindowMessage
AppShow <i>name</i>	Tasks(<i>name</i>). Visible = True
AppShow	Application. Visible = True
AppSize <i>name</i> , <i>width</i> , <i>height</i>	Tasks(<i>name</i>). Resize Width:= <i>width</i> , Height:= <i>height</i>

AppSize <i>width, height</i>	Application. Resize Width:= <i>width</i> , Height:= <i>height</i>
AppWindowHeight <i>name, height</i>	Tasks(<i>name</i>). Height = <i>height</i>
AppWindowHeight <i>height</i>	Application. Height = <i>height</i>
AppWindowPosLeft <i>name, horizpos</i>	Tasks(<i>name</i>). Left = <i>horizpos</i>
AppWindowPosLeft <i>horizpos</i>	Application. Left = <i>horizpos</i>
AppWindowPosTop <i>name, vertpos</i>	Tasks(<i>name</i>). Top = <i>vertpos</i>
AppWindowPosTop <i>vertpos</i>	Application. Top = <i>vertpos</i>
AppWindowWidth <i>name, width</i>	Tasks(<i>name</i>). Width = <i>width</i>
AppWindowWidth <i>width</i>	Application. Width = <i>width</i>
Asc(<i>string</i>)	Asc(<i>string</i>)
AtEndOfDocument()	If Selection.Type = wdSelectionIP and Selection. End : ActiveDocument.Content.End - 1 Then atEnd = True
AtStartOfDocument()	If Selection.Type = wdSelectionIP and Selection. Start atStart = True
AutoMarkIndexEntries	ActiveDocument.Indexes. AutoMarkEntries
AutomaticChange	Application. AutomaticChange
AutoText	Selection.Range. InsertAutoText
AutoTextName\$(<i>num, context</i>)	x = ActiveDocument.AttachedTemplate.AutoTextEntries(



Visual Basic Equivalents B

A B C D E F G H I J K L M N O P R S T U V W Y

B

Beep	Beep
Begin Dialog...End Dialog	Create and display a custom form . For information about controls to a form, see Adding controls to a user form . Selection.Font. Bold = True
Bold, Bold()	x = Selection.Font. Bold
<i>name</i> = BookmarkName\$(<i>num</i>)	<i>name</i> = ActiveDocument.Bookmarks(<i>num</i>). Name With ActiveDocument.Paragraphs(1).Borders(wdBorder LineStyle = wdLineStyleSingle LineWidth = wdLineWidth075pt
BorderBottom, BorderBottom()	End With x = ActiveDocument.Paragraphs(1).Borders(wdBorderBotto With Selection.Borders InsideLineStyle = wdLineStyleSingle InsideLineWidth = wdLineWidth075pt
BorderInside, BorderInside()	End With x = Selection.Borders. InsideLineStyle

	With ActiveDocument.Paragraphs(1).Borders(wdBorder
	. LineStyle = wdLineStyleSingle
BorderLeft, BorderLeft()	. LineWidth = wdLineWidth075pt
	End With
	x =
	ActiveDocument.Paragraphs(1).Borders(wdBorderLeft).
	Selection.Borders(wdBorderType). LineStyle = wdLineStyle
BorderLineStyle, BorderLineStyle()	x =
	ActiveDocument.Paragraphs(1).Borders(wdBorderType)
	Selection.Borders. Enable = False
	' or
BorderNone, BorderNone()	Selection.Borders(wdBorderLeft). LineStyle = wdLineStyle
	x = Selection.Range.Borders. Enable
	With Selection.Borders
	. OutsideLineStyle = wdLineStyleSingle
BorderOutside, BorderOutside()	. OutsideLineWidth = wdLineWidth075pt
	End With
	x = Selection.Borders. OutsideLineStyle
	With ActiveDocument.Paragraphs(1).Borders(wdBorder
	. LineStyle = wdLineStyleSingle
BorderRight, BorderRight()	. LineWidth = wdLineWidth075pt

End With

x =

ActiveDocument.Paragraphs(1).Borders(wdBorderRight

With Selection.Paragraphs(1).Borders(wdBorderTop)

[.LineStyle](#) = wdLineStyleSingle

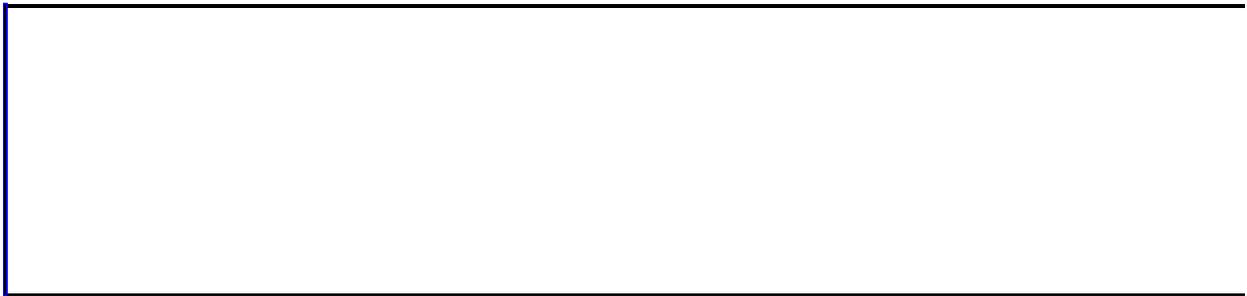
[.LineWidth](#) = wdLineWidth075pt

BorderTop,
BorderTop()

End With

x =

ActiveDocument.Paragraphs(1).Borders(wdBorderTop).



Visual Basic Equivalents C

A B C D E F G H I J K L M N O P R S T U V W Y

C

	Call
	' or
Call	WordBasic.Call
	' or
	Application. Run
	Selection. ColumnSelectMode = False
Cancel	Selection. ExtendMode = False
	Selection. EscapeKey
CancelButton	CommandButton
	Selection.Paragraphs. Alignment = wdAlignPara
CenterPara, CenterPara()	x = Selection.Paragraphs. Alignment
	Selection.Range. Case = <i>WdCharacterCase</i>
ChangeCase, ChangeCase()	x = Selection.Range. Case
	Selection.Font. ColorIndex = <i>WdColorIndex</i>
CharColor, CharColor()	x = Selection.Font. ColorIndex
	Selection. MoveLeft Unit:=wdCharacter, Count:

CharLeft 1	Extend:=wdMove
CharLeft 1, 1	Selection. MoveLeft Unit:=wdCharacter, Count: Extend:=wdExtend
<i>num = CharLeft(1)</i>	<i>num = Selection.MoveLeft(Unit:=wdCharacter,</i>
CharRight 1	Selection. MoveRight Unit:=wdCharacter, Coun Extend:=wdMove
CharRight 1, 1	Selection. MoveRight Unit:=wdCharacter, Coun Extend:=wdExtend
<i>num = CharRight(1)</i>	<i>num = Selection.MoveRight(Unit:=wdCharacte</i>
ChDefaultDir <i>path</i> , <i>wdDefaultFilePath</i>	Options. DefaultFilePath (<i>WdDefaultFilePath</i>) = ChDir <i>path</i>
ChDir <i>path</i>	' or Application. ChangeFileOpenDirectory
CheckBox	CheckBox control
CheckBoxFormField	ActiveDocument.FormFields. Add Range:= <i>rang</i> Type:=wdFieldFormCheckBox With CommandBars(<i>name</i>).Controls(1) .FaceId = <i>num</i> .ToolTipText = <i>text</i>
ChooseButtonImage	End With
Chr\$(<i>num</i>)	Chr(<i>num</i>)
CleanString\$(<i>string</i>)	x = CleanString (<i>string</i>)
ClearAddInst	Addins. Unload
ClearFormField	ActiveDocument.FormFields(1).TextInput. Clea

Close [Close](#)
ClosePane ActiveWindow.ActivePane.[Close](#)
ClosePreview ActiveDocument.[ClosePrintPreview](#)
CloseUpPara Selection.Paragraphs.[CloseUp](#)
CloseViewHeaderFooter ActiveWindow.View.[SeekView](#) = wdSeekMainl
If ActiveDocument.Bookmarks(*name*) =
ActiveDocument.Bookmarks(*name*) Then

 same = True

CmpBookmarks()

End If

The [Start](#) and [End](#) properties can be used to con
and ending positions of two bookmarks.

ColumnSelect

Selection.[ColumnSelectMode](#) = True

ComboBox

ComboBox control

CommandValid()

Use the [IsValid](#) property to determine if a
reference is valid. Also an object variable that r
valid.

Connect

System.[Connect](#)

[WordBasic](#).ControlRun

ControlRun

' or

Shell *appfilename*

Converter\$(*num*)

x = FileConverters(*num*).[ClassName](#)

ConverterLookup(*name*)

x = FileConverters(*name*).[SaveFormat](#)

With ActiveDocument.Shapes(1).OLEFormat

[IconIndex](#) = *num*

[ActivateAs](#) = *text*

ConvertObject *IconNumber*,
ActivateAs, *IconFileName*,

[IconPath](#) & Application.PathSeparator & [Ic](#)

[IconLabel](#) = *text*

Caption, Class, DisplayIcon [.ClassType](#) = text
[.DisplayAsIcon](#) = True

End With

CopyBookmark ActiveDocument.Bookmarks(*name*).[Copy](#)(*name*)
CopyButtonImage CommandBars(*name*).Controls(1).CopyFace
CopyFile FileCopy
CopyFormat Selection.[CopyFormat](#)
CopyText Application.[Run](#) MacroName:="CopyText"
CountAddins() x = Addins.[Count](#)
CountAutoCorrectExceptions(0) x = AutoCorrect.FirstLetterExceptions.[Count](#)

CountAutoCorrectExceptions(1) x = AutoCorrect.TwoInitialCapsExceptions.[Count](#)

CountAutoTextEntries() x = ActiveDocument.AttachedTemplate.AutoTextEntries.[Count](#)
CountBookmarks() x = ActiveDocument.Bookmarks.[Count](#)
myPath = "C:\\"

myName = Dir(myPath, vbDirectory)

Do While myName <> ""

 If myName <> "." And myName <> ".." Then

 If (GetAttr(myPath & myName) And vbDirectory) Then

CountDirectories() count = count + 1

 End If

End If

myName = Dir

```

Loop
MsgBox count & " directories"

x = ActiveDocument.BuiltInDocumentProperties.Count
' or
x = ActiveDocument.CustomDocumentProperties.Count

CountDocumentVars()
CountFiles()
x = ActiveDocument.Variables.Count
x = RecentFiles.Count
x = FontNames.Count
' or

CountFonts()
x = PortraitFontNames.Count
' or
x = LandscapeFontNames.Count

CountFoundFiles()
x = Application.FileSearch.FoundFiles.Count
CustomizationContext = template or document

CountKeys()
x = KeyBindings.Count

CountLanguages()
x = Languages.Count
' no direct equivalent
' counts the number of modules associated with
For Each xItem In NormalTemplate.VBProject.Modules
    If xItem.Type = vbext_ct_StdModule Then Count
Next x

CountMacros()
MsgBox Count

```

CountMenuItems()	x = CommandBars(<i>name</i>).Controls. Count
CountMenus()	x = CommandBars.ActiveMenuBar.Controls. Count
CountMergeFields()	x = ActiveDocument.MailMerge.Fields. Count x = ActiveDocument.Styles. Count
	<i>' or</i>
	x = ActiveDocument.AttachedTemplate.OpenAsDo
CountStyles()	<i>' to exclude built-in styles from the count</i> For Each xSty In ActiveDocument. Styles If xSty. BuiltIn = False Then aCount = aCount + 1 Next xSty
CountToolbarButtons()	x = CommandBars(<i>name</i>).Controls. Count For Each xCB In CommandBars
CountToolbars()	If xCB. Type = msoBarTypeNormal Then aCount = aCount + 1 Next xCB
CountToolsGrammarStatistics()	x = ActiveDocument.Content.ReadabilityStatistics
CountWindows()	x = Windows. Count
CreateSubdocument	ActiveDocument.Subdocuments. AddFromRange



Visual Basic Equivalents D

A B C D E F G H I J K L M N O P R S T U V W Y

D

Date\$()	Date
DateSerial()	DateSerial
DateValue()	DateValue
Day()	Day
Days360()	DateDiff
DDEExecute <i>channel</i> , <i>command</i>	DDEExecute <i>channel</i> , <i>command</i>
<i>chan</i> = DDEInitiate (<i>application</i> , <i>topic</i>)	<i>chan</i> = DDEInitiate (<i>application</i> , <i>topic</i>)
DDEPoke <i>channel</i> , <i>item</i> , <i>data</i>	DDEPoke <i>channel</i> , <i>item</i> , <i>data</i>
<i>data</i> = DDERequest\$(<i>channel</i> , <i>item</i>)	<i>data</i> = DDERequest (<i>channel</i> , <i>item</i>)
DDETerminate <i>channel</i>	DDETerminate <i>channel</i>
DDETerminateAll	DDETerminateAll
Declare	Declare
DefaultDir\$()	<i>x</i> = DefaultFilePath (<i>WdDefaultFilePath</i>)
DeleteAddIn <i>name</i>	Addins(<i>name</i>). Delete
DeleteBackWord	Selection. Delete Unit:=wdWord, Count:=-1
DeleteButton	CommandBars(<i>name</i>).Controls(<i>num</i>). Delete
DeleteDocumentProperty <i>name</i>	ActiveDocument.CustomDocumentProperties(<i>name</i>)
DeleteWord	Selection.Words(1). Delete
DemoteList	Selection.Range.ListFormat. ListOutdent
DemoteToBodyList	Selection.Paragraphs(1). OutlineDemoteToBody

Dialog, Dialog()	Dialogs(<i>WdWordDialog</i>). Show
DialogEditor	ShowVisualBasicEditor = True
Dim	Dim
DisableAutoMacros	WordBasic .DisableAutoMacros
DisableInput	Application. EnableCancelKey = <i>WdEnableCancelK</i>
DlgControlId()	WordBasic dynamic dialog functionality has been replaced by custom user forms. Refer to the topics in Microsoft Forms Help.
DlgEnable, DlgEnable()	
DlgFilePreview,	
DlgFilePreview\$()	
DlgFocus, DlgFocus\$()	
DlgListBoxArray,	
DlgListBoxArray()	
DlgLoadValues,	
DlgLoadValues()	
DlgSetPicture	
DlgStoreValues	
DlgText, DlgText\$()	
DlgUpdateFilePreview	
DlgValue, DlgValue()	
DlgVisible, DlgVisible()	
DocClose	ActiveWindow. Close
DocMaximize,	
DocMaximize()	ActiveWindow. WindowState = wdWindowStateMaximize
DocMinimize,	
DocMinimize()	ActiveWindow. WindowState = wdWindowStateMinimize
	With ActiveWindow
	.Top = <i>VertPos</i>
DocMove <i>HorizPos</i> ,	
<i>VertPos</i>	.Left = <i>HorizPos</i>
	End With
DocRestore	ActiveWindow. WindowState = wdWindowStateNormal

```

DocSize width, height
    With ActiveWindow
        .Height = width
        .Width = height
    End With

DocSplit, DocSplit()
    ActiveWindow.SplitVertical = 50
    x = ActiveWindow.SplitVertical

DocumentHasMisspellings()
    x = ActiveDocument.SpellingErrors.Count
    ' enumerate the DocumentProperties collection

DocumentPropertyExists()
    For Each aProp In
        ActiveDocument.CustomDocumentProperties
            If aProp.Name = name Then itExists = True
    Next aProp

DocumentPropertyName$(num)
    x =
        ActiveDocument.CustomDocumentProperties(num)
    ' or
    x =
        ActiveDocument.BuiltInDocumentProperties(num).

DocumentPropertyType()
    x =
        ActiveDocument.CustomDocumentProperties(name)

DocumentProtection()
    x = ActiveDocument.ProtectionType
    With ActiveDocument
        var1 = .Name
        var2 = .Path
    End With

```

```

var3 =
.BuiltInDocumentProperties(wdPropertyTemplate)

var4 = .BuiltInDocumentProperties(wdPropertyT

var5 =
.BuiltInDocumentProperties(wdPropertyTimeCreat

var6 =
.BuiltInDocumentProperties(wdPropertyTimeLastS

DocumentStatistics
FileName, Directory,
Template, Title, Created,
LastSaved, LastSavedBy,
Revision, Time, Printed,
Pages, Words, Characters,
Paragraphs, Lines, FileSize
var7 =
.BuiltInDocumentProperties(wdPropertyLastAuthor

var8 = .BuiltInDocumentProperties(wdPropertyR

var9 =
.BuiltInDocumentProperties(wdPropertyVBATotalF

var10 =
.BuiltInDocumentProperties(wdPropertyTimeLastP

var11 = .BuiltInDocumentProperties(wdProperty

var12 = .BuiltInDocumentProperties(wdProperty

var13 =
.BuiltInDocumentProperties(wdPropertyCharacters

var14 = .BuiltInDocumentProperties(wdProperty

var15 = .BuiltInDocumentProperties(wdProperty

var16 = .BuiltInDocumentProperties(wdProperty

End With

DocWindowHeight ActiveWindow.Height = height
DocWindowPosLeft ActiveWindow.Left = horizpos
DocWindowPosTop ActiveWindow.Top = vertpos

```

DocWindowWidth	ActiveWindow. Width = <i>width</i>
DoFieldClick	Selection.Fields(1). DoClick
DOSToWin\$()	x = WordBasic . [DOSToWin\$] (<i>StringToTranslate</i>)
DottedUnderline, DottedUnderline()	Selection.Font. UnderLine = wdUnderlineDotted x = Selection.Font. UnderLine
DoubleUnderline, DoubleUnderline()	Selection.Font. UnderLine = wdUnderlineDouble x = Selection.Font. UnderLine
Drawing object statements and functions	Use the properties and methods of the following objects: Shape , Shapes , and ShapeRange .
DropDownFormField	ActiveDocument.FormFields. Add Range:= <i>range</i> , Type:=wdFieldFormDropDown
DropListBox	ComboBox control



Visual Basic Equivalents E

A B C D E F G H I J K L M N O P R S T U V W Y

E

EditAutoText .Name = <i>name</i> , .Add	ActiveDocument.AttachedTemplate.AutoTextEntries(<i>name</i>).Add
EditAutoText .Name = <i>name</i> , .InsertAs = 0, .Insert	ActiveDocument.AttachedTemplate.AutoTextEntries(<i>name</i>).Insert Where:= <i>range</i> , RichText:=True
EditAutoText .Name = <i>name</i> , .Delete	Templates(<i>name</i>).AutoTextEntries(<i>name</i>). Delete
EditBookmark .Name = <i>name</i> , .Add	ActiveDocument.Bookmarks. Add Name:= <i>name</i> , Range:= <i>range</i>
EditBookmark .Name = <i>name</i> , .Delete	ActiveDocument.Bookmarks(<i>name</i>). Delete
EditBookmark .Name = <i>name</i> , .Goto	ActiveDocument.Bookmarks(<i>name</i>). Select
EditBookmark .Name = <i>name</i> , .SortBy	ActiveDocument.Bookmarks. DefaultSorting = wdSortBy
EditButtonImage	WordBasic .EditButtonImage
EditClear	Selection.Range. Delete
EditConvertAllEndnotes	ActiveDocument.Endnotes. Convert
EditConvertAllFootnotes	ActiveDocument.Footnotes. Convert
	Selection.Footnotes. Convert
EditConvertNotes	' or

	Selection.Endnotes. Convert
EditCopy	Selection.Range. Copy
EditCopyAsPicture	Selection.Range. CopyAsPicture
EditCreatePublisher	Selection.Range. CreatePublisher
EditCut	Selection.Range. Cut
EditFind	Selection. Find
EditFindBorder	Selection. Find .Borders
EditFindClearFormatting	Selection.Find. ClearFormatting
EditFindFont	Selection.Find. Font
EditFindFound()	Selection.Find. Found
EditFindFrame	Selection.Find. Frame
EditFindHighlight	Selection.Find. Highlight = True
EditFindLang	Selection. Find .LanguageID
EditFindNotHighlight	Selection.Find. Highlight = False
EditFindPara	Selection.Find. ParagraphFormat
EditFindStyle	Selection.Find. Style
EditFindTabs	Selection.Find.ParagraphFormat. TabStops
EditGoTo	Selection. Goto
	ActiveDocument.Shapes(1).OLEFormat. Open

' or

ActiveDocument.InlineShapes(1).OLEFormat.[Open](#)

With ActiveDocument.InlineShapes(1).LinkFormat

.[AutoUpdate](#) = True

.[Locked](#) = True

.[SavePictureWithDocument](#) = True

.[Update](#)

.[BreakLink](#)

*EditLinks UpdateMode,
Locked, SavePictureInDoc,
UpdateNow, OpenSource,
KillLink, Link, Application,
Item, FileName*

	.Application.Name
	.SourceFullName
	End With
	Selection.InlineShapes(1).OLEFormat. Edit
EditObject	' or Selection.ShapeRange(1).OLEFormat. Edit
EditPaste	Selection.Range. Paste
EditPasteSpecial	Selection.Range. PasteSpecial
EditPicture	Selection.ShapeRange(1). Activate
EditPublishOptions	ActiveDocument. EditionOptions
EditRedo	ActiveDocument. Redo
EditRepeat	Repeat
EditReplaceBorder	Selection.Find. Replacement .Borders
EditReplaceClearFormatting	Selection.Find.Replacement. ClearFormatting
EditReplaceFont	Selection.Find. Replacement .Font
EditReplaceFrame	Selection.Find. Replacement .Frame
EditReplaceHighlight	Selection.Find.Replacement. Highlight = True
EditReplaceLang	Selection.Find. Replacement .LanguageID
EditReplaceNotHighlight	Selection.Find.Replacement. Highlight = False
EditReplacePara	Selection.Find.Replacement. ParagraphFormat
EditReplaceStyle	Selection.Find.Replacement. Style
EditReplaceTabs	Selection.Find.Replacement. ParagraphFormat .TabS ActiveDocument.Content. Select
EditSelectAll	' or Selection. WholeStory
EditSubscribeOptions	ActiveDocument. EditionOptions
EditSubscribeTo	Selection.Range. SubscribeTo

	ActiveDocument.Endnotes. SwapWithFootnotes
EditSwapAllNotes	' or ActiveDocument.Footnotes. SwapWithEndnotes
EditTOACategory	ActiveDocument.TablesOfAuthoritiesCategories(<i>name</i>)
EditUndo	ActiveDocument. Undo
EmptyBookmark(<i>name</i>)	x = ActiveDocument.Bookmarks(<i>name</i>). Empty
EnableFormField	ActiveDocument.FormFields(<i>name</i>). Enabled = True
EndOfColumn, EndOfColumn()	Selection. EndOf Unit:=wdColumn, Extend:=wdMove
EndOfDocument, EndOfDocument()	Selection. EndKey Unit:=wdStory
EndOfLine, EndOfLine()	Selection. EndKey Unit:=wdLine, Extend:=wdMove
EndOfRow, EndOfRow()	Selection. EndKey Unit:=wdRow, Extend:=wdMove
EndOfWindow, EndOfWindow()	Selection. MoveDown Unit:=wdWindow
Environ\$()	Environ\$()
Eof()	EOF()
Err	Err
Error	Error
ExistingBookmark(<i>name</i>)	x = ActiveDocument.Bookmarks. Exists (<i>name</i>)
ExitWindows	Tasks. ExitWindows
ExtendMode()	x= Selection. ExtendMode ' <i>activates extend mode</i> Selection. ExtendMode = True
ExtendSelection	' <i>extends the selection</i> Selection. Expand Unit:=wdUnits



Visual Basic Equivalents F

A B C D E F G H I J K L M N O P R S T U V W Y

F

FieldSeparator\$	Application. DefaultTableSeparator
FileAOCEAddMailer	ActiveDocument. HasMailer = True
FileAOCEDeleteMailer	ActiveDocument. HasMailer = False
FileAOCEExpandMailer	Macintosh only
FileAOCEForwardMail	ActiveDocument. ForwardMailer
FileAOCENextLetter	Application. NextLetter
FileAOCEReplyAllMail	Macintosh only
FileAOCEReplyMail	Macintosh only
FileAOCESendMail	ActiveDocument. SendMailer
FileClose	ActiveDocument. Close
FileCloseAll	Documents. Close
FileClosePicture	ActiveDocument. Close
FileConfirmConversions	Options. ConfirmConversions = True
FileCreator\$()	Macintosh only
FileDocumentLayout	Macintosh only
FileExit	Application. Quit
FileFind	Application. FileSearch
FileList <i>number</i>	RecentFiles(<i>num</i>). Open
FileMacCustomPageSetupGX	Macintosh only
FileMacPageSetup	Macintosh only
FileMacPageSetupGX	Macintosh only
FileName\$()	x = ActiveDocument. FullName
FileName\$(<i>num</i>)	x = RecentFiles(<i>num</i>). Name & Application.PathSeparator RecentFiles(<i>num</i>). Path

FileNameFromWindow\$()	x = Windows(1).Document. FullName x = WordBasic . [FileNameInfo\$]()
	x = ActiveDocument. Name
FileNameInfo\$()	x = ActiveDocument. Path
	x = ActiveDocument. FullName
FileNew <i>Template</i>	Documents. Add <i>Template:=filename</i>
FileNewDefault	Documents. Add
File <i>num</i>	RecentFiles(<i>num</i>). Open
FileOpen	Documents. Open
FilePost	ActiveDocument. Post
	With ActiveDocument.PageSetup
	.TopMargin = <i>num</i>
	.BottomMargin = <i>num</i>
	.LeftMargin = <i>num</i>
	.RightMargin = <i>num</i>
	.Gutter = <i>num</i>
	.PageHeight = <i>num</i>
	.PageWidth = <i>num</i>
	.Orientation = <i>WdOrientation</i>
	.FirstPageTray = <i>WdPaperTray</i>
	.OtherPagesTray = <i>WdPaperTray</i>
FilePageSetup <i>Tab</i> ,	.VerticalAlignment = <i>WdVerticalAlignment</i>
<i>TopMargin</i> , <i>BottomMargin</i> ,	.SetAsTemplateDefault
<i>LeftMargin</i> , <i>RightMargin</i> ,	
<i>Gutter</i> , <i>PageWidth</i> ,	
<i>PageHeight</i> , <i>Orientation</i> ,	

*FirstPage, OtherPages,
VertAlign, ApplyPropsTo,
Default, FacingPages,
HeaderDistance,
FooterDistance, SectionStart,
OddAndEvenPages,
DifferentFirstPage, Endnotes,
LineNum, StartingNum,
FromText, CountBy,
NumMode*

[.MirrorMargins](#) = True
[.HeaderDistance](#) = *num*
[.FooterDistance](#) = *num*
[.SectionStart](#) = *WdSectionStart*
[.OddAndEvenPagesHeaderFooter](#) = True
[.DifferentFirstPageHeaderFooter](#) = True
[.SuppressEndnotes](#) = True

With LineNumbering

[.Active](#) = True
[.StartingNumber](#) = *num*
[.DistanceFromText](#) = *num*
[.CountBy](#) = *num*
[.RestartMode](#) = *WdNumberingRule*

End With

End With

FilePreview
FilePrint
FilePrintDefault
FilePrintOneCopy

FilePrintPreview,
FilePrintPreview()

Image control
ActiveDocument.[PrintOut](#)
ActiveDocument.[PrintOut](#)
Macintosh only
[PrintPreview](#) = True

x = [PrintPreview](#)

[PrintPreview](#) = True

```

FilePrintPreviewFullScreen    ActiveWindow.View.FullScreen = True

                                With ActiveWindow.View.Zoom

                                    .PageColumns = 2
FilePrintPreviewPages,      .PageRows = 1
FilePrintPreviewPages()

                                End With

FilePrintSetup              ActivePrinter
FileProperties              WordBasic.FileProperties
FileQuit                   Application.Quit
                                With ActiveDocument.RoutingSlip

                                    .Subject = text

                                    .Message = text
FileRoutingSlip Subject,    .Delivery = WdRoutingSlipDelivery
FileRoutingSlip Message, AllAtOnce,
FileRoutingSlip ReturnWhenDone, .ReturnWhenDone = True
FileRoutingSlip TrackStatus, Protect

FileRoutingSlip AddSlip    .TrackStatus = True

FileRoutingSlip            .Protect = WdProtectionType
FileRoutingSlip RouteDocument

                                End With
FileRoutingSlip AddRecipient ActiveDocument.HasRoutingSlip = True
FileRoutingSlip ResetSlip   ActiveDocument.Route
FileRoutingSlip ClearSlip  ActiveDocument.RoutingSlip.AddRecipient

                                ActiveDocument.RoutingSlip.Reset

                                ActiveDocument.HasRoutingSlip = False

```

Files\$()
FileSave
FileSaveAll
FileSaveAs
FileSendMail

Dir()
ActiveDocument.[Save](#)
Documents.[Save](#)
ActiveDocument.[SaveAs](#)
ActiveDocument.[SendMail](#)
With ActiveDocument

[.BuiltInDocumentProperties](#)(wdPropertyTitle)

[.BuiltInDocumentProperties](#)(wdPropertySubject)

[.BuiltInDocumentProperties](#)(wdPropertyLastAuthor)

[.BuiltInDocumentProperties](#)(wdPropertyKeywords)

[.BuiltInDocumentProperties](#)(wdPropertyComments)

[.Name](#)

[.Path](#)

FileSummaryInfo *Title,*
Subject, Author, Keywords,
Comments, FileName,
Directory, Template,
CreateDate, LastSavedDate,
LastSavedBy,
RevisionNumber, EditTime,
LastPrintedDate, NumPages,
NumWords, NumChars,
NumParas, NumLines,
FileSize

[.BuiltInDocumentProperties](#)(wdPropertyTemplate)

[.BuiltInDocumentProperties](#)(wdPropertyTimeCreated)

[.BuiltInDocumentProperties](#)(wdPropertyTimeLastSaved)

[.BuiltInDocumentProperties](#)(wdPropertyLastAuthor)

[.BuiltInDocumentProperties](#)(wdPropertyRevisionNumber)

[.BuiltInDocumentProperties](#)(wdPropertyVBATags)

[.BuiltInDocumentProperties](#)(wdPropertyTimeLastPrinted)

[.BuiltInDocumentProperties](#)(wdPropertyPages)

[.BuiltInDocumentProperties](#)(wdPropertyWords)

[.BuiltInDocumentProperties](#)(wdPropertyCharacterCount)

.BuiltInDocumentProperties(wdPropertyParas)

.BuiltInDocumentProperties(wdPropertyLines)

.BuiltInDocumentProperties(wdPropertyBytes)

End With

FileTemplates

ActiveDocument.[AttachedTemplate](#) = *template*

FileType\$()

Macintosh only

Selection.Font.[Name](#) = *text*

Font, Font\$()

x = Selection.Font.[Name](#)

Selection.Font.[Size](#) = *num*

FontSize, FontSize()

x = Selection.Font.[Size](#)

FontSizeSelect

Application.[Run](#) MacroName:="FontSizeSelect"

FontSubstitution

Application.[SubstituteFont](#)

For...Next

For...Next

' *Set properties of the Font object*

With ActiveDocument.Envelope.Address.[Font](#)

.[Size](#) = *num*

FormatAddrFonts

.[ColorIndex](#) = *WdColorIndex*

.[Bold](#) = True

End With

FormatAutoFormat

ActiveDocument.[AutoFormat](#)

With ActiveDocument.Paragraphs(1).Borders

.[Shadow](#) = True

.[DistanceFromBottom](#) = *num*

[.DistanceFromTop](#) = num

[.DistanceFromLeft](#) = num

[.DistanceFromRight](#) = num

End With

With Selection.Shading

FormatBordersAndShading

Shadow, TopBorder,

LeftBorder, BottomBorder,

RightBorder, HorizBorder,

VertBorder, TopColor,

LeftColor, BottomColor,

RightColor, HorizColor,

VertColor, FineShading,

FromText, Shading,

Foreground, Background, Tab

[.Texture](#) = WdTextureIndex

[.BackgroundPatternColorIndex](#) = WdColorIndex

[.ForegroundPatternColorIndex](#) = WdColorIndex

End With

With ActiveDocument.Paragraphs(1)

[.Borders\(WdBordertype\).LineStyle](#) = WdLineStyle

[.Borders\(WdBordertype\).LineWidth](#) = WdLineWidth

[.Borders\(WdBordertype\).ColorIndex](#) = WdColorIndex

End With

With Dialogs(wdDialogFormatBordersAndShading)

[.DefaultTab](#) = WdWordDialogTab

[.Show](#)

End With

With ListGalleries(wdBulletGallery).ListTemplate

[.NumberFormat](#) = ChrW(num)

FormatBullet *Points, Color, Alignment, Indent, Space, Hang, CharNum, Font*

[.NumberStyle](#) = wdListNumberStyleBullet

[.NumberPosition](#) = *num*

[.Alignment](#) = *WdListLevelAlignment*

[.TextPosition](#) = *num*

[.TabPosition](#) = *num*

With [.Font](#)

[.Size](#) = *num*

[.Name](#) = *text*

[.ColorIndex](#) = *WdColorIndex*

End With

End With

FormatBulletDefault,
FormatBulletDefault()

Selection.Range.ListFormat.[ApplyBulletDefault](#)

Selection.Range.ListFormat.[RemoveNumbers](#)

FormatBulletsAndNumbering
Remove, Hang, Preset

Selection.Range.ListFormat.[ApplyListTemplate](#)
ListTemplate:=ListGalleries(*WdListGalleryType*).

Selection.Range.ListFormat.[RemoveNumbers](#)

With ActiveDocument.Shapes(1).Callout

[.Type](#) = *MsoCalloutType*

[.Gap](#) = *num*

[.Angle](#) = *MsoCalloutAngleType*

[.Drop](#) = *num*

FormatCallout Type, Gap,
Angle, Drop, Length, Border,
AutoAttach, Accent

[.DropType](#) = *MsoCalloutDropType*

[.Length](#) = *num*

[.Border](#) = *MsoTriState*

[.AutoAttach](#) = *MsoTriState*

[.Accent](#) = *MsoTriState*

End With

FormatChangeCase

Selection.Range.[Case](#) = *WdCharacterCase*
With ActiveDocument.[TextColumns](#)

[.SetCount](#) NumColumns:=*num*

[.Width](#) = *num*

FormatColumns *Columns*,
ColumnWidth,
ColumnSpacing,
EvenlySpaced, *ColLine*

[.Spacing](#) = *num*

[.EvenlySpaced](#) = *False*

[.LineBetween](#) = *False*

End With

' Set properties of the [Borders](#) object

With ActiveDocument.Styles(*name*).[Borders](#)

FormatDefineStyleBorders

[.Enable](#) = *True*

[.Shadow](#) = *True*

End With

' Set properties of the [Font](#) object

With ActiveDocument.Styles(*name*).[Font](#)

```

FormatDefineStyleFont      .Bold = True
                           .Name = "Arial"
                           End With

                           ' Set properties of the Frame object
                           With ActiveDocument.Styles(name).Frame

FormatDefineStyleFrame    .Width = num
                           .VerticalPosition = num
                           End With

FormatDefineStyleLang     ActiveDocument.Styles(name).LanguageID = Wd
                           ' Set properties of the ListLevel object
                           With ActiveDocument.Styles(name).
                           ListGalleries(WdListGalleryType).ListTemplates(i

FormatDefineStyleNumbers  .NumberFormat = "%1)"
                           .TrailingCharacter = wdTrailingTab
                           .NumberStyle = wdListNumberStyleArabic
                           End With

                           ' Set properties of the ParagraphFormat object
                           With ActiveDocument.Styles(name).ParagraphFo

FormatDefineStylePara     .SpaceAfter = num
                           .RightIndent = num
                           End With

```

' Set properties of the [TabStops](#) object

FormatDefineStyleTabs

ActiveDocument.Styles(*name*).ParagraphFormat.
[WdTabLeader](#)

FormatDrawingObject

Set properties of the [Shape](#) object.
With ActiveDocument.Paragraphs(1).DropCap

[.Position](#) = *WdDropPosition*

FormatDropCap *Position,*
Font, DropHeight,
DistFromText

[.FontName](#) = *text*

[.LinesToDrop](#) = *num*

[.DistanceFromText](#) = *num*

End With

With Selection.Font

[.Size](#) = *num*

[.Underline](#) = True

[.ColorIndex](#) = *WdColorIndex*

[.StrikeThrough](#) = True

[.Superscript](#) = *num*

[.Subscript](#) = *num*

[.Shadow](#) = True

[.Hidden](#) = True

[.SmallCaps](#) = True

FormatFont *Points,*
Underline, Color,
Strikethrough, Superscript,

[.AllCaps](#) = True

[.Outline](#) = True

*Subscript, Shadow, Hidden,
SmallCaps, AllCaps, Outline,
Spacing, Position, Kerning,
KerningMin, Default, Tab,
Font, Bold, Italic*

[.Spacing](#) = *num*

[.Position](#) = *num*

[.Kerning](#) = *num*

[.SetAsTemplateDefault](#)

[.Font](#) = *name*

[.Bold](#) = True

[.Italic](#) = True

End With

With Dialogs(wdDialogFormatFont)

[.DefaultTab](#) = *WdWordDialogTab*

[.Show](#)

End With

With ActiveDocument.Frames(1)

[.TextWrap](#) = True

[.WidthRule](#) = *WdFrameSizeRule*

[.Width](#) = *num*

[.Height](#) = *num*

[.HeightRule](#) = *WdFrameSizeRule*

[.HorizontalPosition](#) = *num*

[.RelativeHorizontalPosition](#) = *WdRelativeHoriz*

*FormatFrame Wrap,
WidthRule, FixedWidth,
HeightRule, FixedHeight,
PositionHorz,
PositionHorzRel,
DistFromText, PositionVert,*

*PositionVertRel,
DistVertFromText,
MoveWithText, LockAnchor,
RemoveFrame*

[.HorizontalDistanceFromText](#) = num

[.VerticalPosition](#) = num

[.RelativeVerticalPosition](#) = WdRelativeVertical

[.VerticalDistanceFromText](#) = num

[.LockAnchor](#) = True

[.Delete](#)

End With

ActiveDocument.Sections(num).Headers(WdHead
= True

FormatHeaderFooterLink

' or

ActiveDocument.Sections(num).Footers(WdHead
= True

With ListGalleries(WdListGalleryType).ListTempl

FormatHeadingNumber

' Set properties of the ListLevel object and use t

End With

' Set properties of the ListLevel object and use the

Set atemp = ListGalleries(wdOutlineNumberGalle

With atemp.ListLevels(1)

.NumberFormat = "Chapter %1"

FormatHeadingNumbering

.TrailingCharacter = wdTrailingNone

.NumberStyle = wdListNumberStyleArabic

End With

FormatMultilevel

Selection.Range.ListFormat.[ApplyListTemplate](#) L

' Set properties of the ListLevel object and use the

Set atemp = ListGalleries(wdOutlineNumberGalle

atemp.ListLevels(1).NumberStyle = wdListNumb

Selection.Range.ListFormat.[ApplyListTemplate](#) L

' Set properties of the ListLevel object and use the

Set atemp = ListGalleries(wdNumberGallery).List

With atemp.ListLevels(1)

.NumberFormat = "%1."

FormatNumber

.TrailingCharacter = wdTrailingTab

.NumberStyle = wdListNumberStyleArabic

End With

Selection.Range.ListFormat.[ApplyListTemplate](#) L

FormatNumberDefault,
FormatNumberDefault()

Selection.Range.ListFormat.[ApplyNumberDefault](#)

Selection.Range.ListFormat.[RemoveNumbers](#)

With Section.Footers(wdHeaderFooterPrimary).Pa

.[IncludeChapterNumber](#) = True

.[RestartNumberingAtSection](#) = True

FormatPageNumber
ChapterNumber, NumRestart,
NumFormat, StartingNum,
Level, Separator

.[NumberStyle](#) = WdPageNumberStyle

.[StartingNumber](#) = num

[.HeadingLevelForChapter](#) = *num*

[.ChapterPageSeparator](#) = *WdSeparatorType*

End With

With ActiveDocument.Paragraphs(1)

[.LeftIndent](#) = *num*

[.RightIndent](#) = *num*

[.SpaceBefore](#) = *num*

[.SpaceAfter](#) = *num*

[.LineSpacingRule](#) = *WdLineSpacing*

[.LineSpacing](#) = *num*

[.Alignment](#) = *WdParagraphAlignment*

*FormatParagraph LeftIndent,
RightIndent, Before, After,
LineSpacingRule,
LineSpacing, Alignment,
WidowControl,
KeepWithNext, KeepTogether,
PageBreak, NoLineNum,
DontHyphen, Tab, FirstIndent*

[.WidowControl](#) = True

[.KeepWithNext](#) = True

[.KeepTogether](#) = True

[.PageBreakBefore](#) = True

[.NoLineNumber](#) = True

[.Hyphenation](#) = True

[.FirstLineIndent](#) = *num*

End With

With Dialogs(wdDialogFormatParagraph)

[.DefaultTab](#) = *WdWordDialogTab*

[.Show](#)

End With

With ActiveDocument.InlineShapes(1)

[.Width](#) = *num*

[.Height](#) = *num*

[.ScaleHeight](#) = *num*

[.ScaleWidth](#) = *num*

FormatPicture *SetSize,*
CropLeft, CropRight,
CropTop, CropBottom,
ScaleX, ScaleY, SizeX, SizeY

With .PictureFormat

[.CropBottom](#) = *num*

[.CropLeft](#) = *num*

[.CropRight](#) = *num*

[.CropTop](#) = *num*

End With

End With

' Set properties of the Font object

With ActiveDocument.Envelope.ReturnAddress.F

[.Size](#) = *num*

FormatRetAddrFonts

[.ColorIndex](#) = *WdColorIndex*

[.Bold](#) = True

End With

With ActiveDocument.PageSetup

[.VerticalAlignment](#) = *WdVerticalAlignment*

[.SectionStart](#) = *WdSectionStart*

[.SuppressEndnotes](#) = True

FormatSectionLayout
SectionStart, VertAlign,
Endnotes, LineNum,
StartingNum, FromText,
CountBy, NumMode

With LineNumbering

[.Active](#) = True

[.StartingNumber](#) = *num*

[.DistanceFromText](#) = *num*

[.CountBy](#) = *num*

[.RestartMode](#) = *WdNumberingRule*

End With

End With

With ActiveDocument.Styles(*name*)

[.Delete](#)

[.NameLocal](#) = *name*

[.BaseStyle](#) = *text*

[.NextParagraphStyle](#) = *style*

x = [.Type](#)

FormatStyle *Name, Delete,*
Merge, NewName, BasedOn,
NextStyle, Type, FileName,
Source, AddToTemplate,
Define, Rename, Apply

End With

Application.[OrganizerCopy](#)

With ActiveDocument

[.UpdateStyles](#)

[.CopyStylesFromTemplate](#)

End With

ActiveDocument.Styles.[Add](#)

Selection.[Style](#) = *name*

FormatStyleGallery

ActiveDocument.[CopyStylesFromTemplate](#)

With Selection.Paragraphs.TabStops

[.ClearAll](#)

FormatTabs *Position,*
DefTabs, Align, Leader, Set,
Clear, ClearAll

[.Add](#) Position:=*num*, Alignment:= *WdTabAlign*

.Item(1).[Clear](#)

End With

ActiveDocument.[DefaultTabStop](#)

With ActiveDocument.FormFields(1)

[.EntryMacro](#) = *text*

[.ExitMacro](#) = *text*

[.Name](#) = *text*

[.Enabled](#) = True

[.OwnHelp](#) = True

[.HelpText](#) = *text*

[.OwnStatus](#) = True

[.StatusText](#) = *text*

```
FormFieldOptions Entry,      .Type = WdFieldType
Exit, Name, Enable, TextType,
TextWidth, TextDefault,     End With
TextFormat, CheckSize,     With ActiveDocument.FormFields(1).TextInput
CheckWidth, CheckDefault,
Type, OwnHelp, HelpText,   .Width = num
OwnStat, StatText          .Default = text

                             .EditType

End With

With ActiveDocument.FormFields(1).CheckBox

    .Size = num

    .AutoSize = True

    .Default = True

End With
```

```
FormShading                 ActiveDocument.FormFields.Shaded = True
FoundFileName$()           Application.FileSearch.FoundFiles (num)
Function...End Function    Function...End Function
```

Visual Basic Equivalents G

A B C D E F G H I J K L M N O P R S T U V W Y

G

GetAddInID(<i>name</i>)	x = Addins(<i>name</i>). Index
GetAddInName\$(<i>num</i>)	x = Addins(<i>num</i>). Name
GetAddress\$()	x = Application. GetAddress
GetAttr(<i>filename</i>)	GetAttr(<i>filename</i>)
GetAutoCorrect\$(<i>name</i>)	x = AutoCorrect.Entries(<i>name</i>). Value x = AutoCorrect.FirstLetterExceptions(<i>num</i>). Name
GetAutoCorrectException\$()	x = AutoCorrect.TwoInitialCapsExceptions(<i>num</i>)
GetAutoText\$()	x = ActiveDocument.AttachedTemplate.AutoTextEnt
GetBookmark\$(<i>name</i>)	x = ActiveDocument.Bookmarks(<i>name</i>).Range. Text
GetCurValues	Dialogs(<i>WdWordDialog</i>). Update
GetDirectory\$()	x = WordBasic . [GetDirectory\$]() x = ActiveDocument.CustomDocumentProperties
GetDocumentProperty(), GetDocumentProperty\$()	' or x = ActiveDocument.BuiltInDocumentProperties
GetDocumentVar\$(<i>name</i>)	x = ActiveDocument.Variables(<i>name</i>). Value
GetDocumentVarName\$(<i>num</i>)	x = ActiveDocument.Variables(<i>num</i>). Name
GetFieldData\$()	x = Selection.Fields(1). Data
GetFormResult(), GetFormResult\$()	x = ActiveDocument.FormFields(<i>name</i>). Result
GetMergeField\$()	x = ActiveDocument.MailMerge.DataSource.DataFile

GetPrivateProfileString\$(<i>filename</i> , <i>section</i>)	x = System. PrivateProfileString (<i>filename</i> , <i>section</i>)
GetProfileString\$(<i>section</i> , <i>key</i>)	x = System. ProfileString (<i>section</i> , <i>key</i>)
GetSelEndPos()	x = Selection. End
GetSelStartPos()	x = Selection. Start
GetSystemInfo\$(21)	x = System. OperatingSystem
GetSystemInfo\$(22)	x = System. ProcessorType
GetSystemInfo\$(23)	' not available
GetSystemInfo\$(24)	x = System. Version
GetSystemInfo\$(25)	' not available
GetSystemInfo\$(26)	x = System. FreeDiskSpace
GetSystemInfo\$(27)	' not available
GetSystemInfo\$(28)	x = System. MathCoprocesorInstalled
GetSystemInfo\$(29)	x = System. Country
GetSystemInfo\$(30)	x = System. LanguageDesignation
GetSystemInfo\$(31)	x = System. VerticalResolution
GetSystemInfo\$(32)	x = System. HorizontalResolution
GetSystemInfo\$(32)	Values 512 to 526 are Macintosh only.
GetText\$(<i>Pos1</i> , <i>Pos2</i>)	x = ActiveDocument. Range (<i>Pos1</i> , <i>Pos2</i>).Text
GoBack	Application. GoBack
Goto	GoTo
GoToAnnotationScope	Selection.Comments(1). Scope .Select
	If Selection.HeaderFooter.IsHeader = True Then
	ActiveWindow.ActivePane.View. SeekView =
	wdSeekCurrentPageFooter

GoToHeaderFooter	Else ActiveWindow.ActivePane.View.SeekView = wdSeekCurrentPageHeader End If
GoToNextAnnotation	Selection. GoToNext (wdGoToComment)
GoToNextEndnote	Selection. GoToNext (wdGoToEndnote)
GoToNextFootnote	Selection. GoToNext (wdGoToFootnote)
GoToNextPage	Selection. GotoNext (wdGoToPage)
GoToNextSection	Selection. GotoNext (wdGoToSection)
GoToNextSubdocument	Selection. NextSubdocument
GoToPreviousItem	Selection. GoTo What:=WdGoToItem, Which:=wd
GroupBox	Frame control
GrowFont	Selection.Font. Grow
GrowFontOnePoint	Selection.Font. Size = Selection.Font.Size + 1

Visual Basic Equivalents H

A B C D E F G H I J K L M N O P R S T U V W Y

H

HangingIndent	ActiveDocument.Paragraphs(1). TabHangingInden
Help	Assistant. Help
HelpAbout	Application. Help HelpType:=wdHelpAbout
HelpActiveWindow	Application. Help HelpType:=wdHelpActiveWindow
HelpContents	Application. Help HelpType:=wdHelpContents
HelpExamplesAndDemos	Not applicable in Word 2002
HelpIndex	Application. Help HelpType:=wdHelpIndex
HelpKeyboard	Not applicable in Word 2002
HelpMSN	Not applicable in Word 2002
HelpPSSHelp	Application. Help HelpType:=wdHelpPSSHelp
HelpQuickPreview	Not applicable in Word 2002
HelpSearch	Application. Help HelpType:=wdHelpSearch
HelpTipOfTheDay	Assistant.FeatureTips = True
HelpTool	Application. HelpTool
HelpUsingHelp	Application. Help HelpType:=wdHelpUsingHelp
HelpWordPerfectHelp	Options. WPHelp = True
HelpWordPerfectHelpOptions	Options. SetWPHelpOptions
Hidden	Selection.Font. Hidden = True Selection.Range. HighlightColorIndex = <i>WdColorIndex</i>
Highlight, Highlight()	x = Selection.Range. HighlightColorIndex
HighlightColor,	Selection.Range. HighlightColorIndex = <i>WdColorIndex</i>

HighlightColor()

x = Selection.Range.[HighlightColorIndex](#)

ActiveWindow.[SmallScroll](#) ToRight:=num

HLine

' or

ActiveWindow.[SmallScroll](#) ToLeft:=num

Hour()

Hour

ActiveWindow.[LargeScroll](#) ToRight:=num

HPage

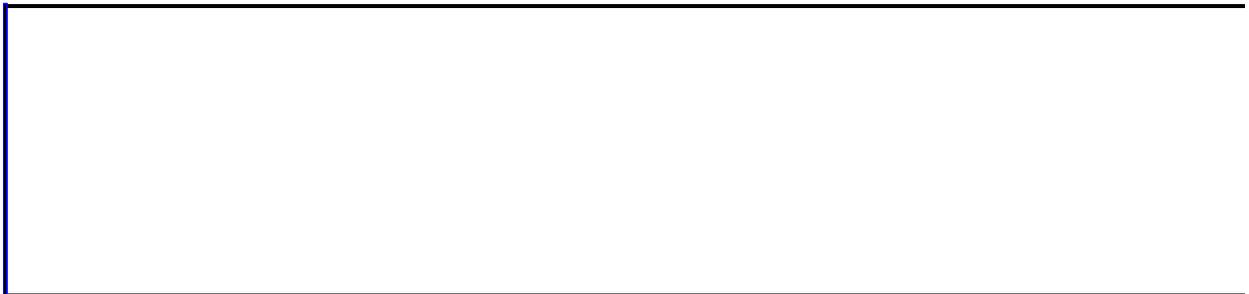
' or

ActiveWindow.[LargeScroll](#) ToLeft:=num

ActiveWindow.[HorizontalPercentScrolled](#) = num

HScroll, HScroll()

num = ActiveWindow.[VerticalPercentScrolled](#)



Visual Basic Equivalents I

A B C D E F G H I J K L M N O P R S T U V W Y

I

If...Then...Else

If...Then...Else

Indent

ActiveDocument.Paragraphs(1).[TabIndent](#)

Input

Input

Input\$()

Input()

InputBox\$()

InputBox

Selection.[InsertAfter](#) Text:=*text*

Insert

' or

Selection.[TypeText](#) Text:=*text*

InsertAddCaption

CaptionLabels.[Add](#)

InsertAddress

Application.[GetAddress](#)

InsertAnnotation

ActiveDocument.Comments.[Add](#)

With AutoCaptions(*name*)

.[AutoInsert](#) = True

.CaptionLabel.[Name](#) = *text*

.CaptionLabel.[Position](#) = *WdCaptionPosition*

InsertAutoCaption *Clear,*
ClearAll, Label, Position

End With

AutoCaptions.[CancelAutoInsert](#)

InsertAutoText

Selection.Range.[InsertAutoText](#)

InsertBreak

Selection.[InsertBreak](#) Type:=*WdBreakType*

InsertCaption	Selection. InsertCaption With CaptionLabels(<i>name</i>) .ChapterStyleLevel = <i>num</i>
InsertCaptionNumbering <i>Label</i> , <i>FormatNumber</i> , <i>ChapterNumber</i> , <i>Level</i> , <i>Separator</i>	.Separator = <i>WdSeparatorType</i> .NumberStyle = <i>WdCaptionNumberStyle</i> .IncludeChapterNumber = True End With
InsertChart	ActiveDocument.Shapes. AddOLEObject
InsertColumnBreak	Selection. InsertBreak Type:=wdColumnBreak
InsertCrossReference	Selection. InsertCrossReference
InsertDatabase	Selection.Range. InsertDatabase
InsertDateField	Selection.Fields. Add Range:= <i>range</i> , Type:=wdl
InsertDateTime	Selection. InsertDateTime
InsertDrawing	ActiveDocument.Shapes. AddOLEObject
InsertEquation	ActiveDocument.Shapes. AddOLEObject
InsertExcelTable	ActiveDocument.Shapes. AddOLEObject
InsertField <i>field_type</i>	ActiveDocument.Fields. Add Range:= <i>range</i> , Ty
InsertFieldChars	Selection.Fields. Add Range:= <i>range</i> , Type:=wdl PreserveFormatting:=False
InsertFile <i>Name</i> , <i>Range</i> , <i>ConfirmConversions</i> , <i>Link</i>	Selection. InsertFile ActiveDocument.Footnotes. Add Range:= <i>range</i> ,
InsertFootnote <i>Reference</i> , <i>NoteType</i>	ActiveDocument.Endnotes. Add Range:= <i>range</i> ,
	Set myField = ActiveDocument.FormFields.Ad Type:= <i>WdFieldType</i>) With myField .EntryMacro = <i>text</i>

InsertFormField *Entry, Exit,*
Name, Enable, TextType,
TextDefault, TextWidth,
TextFormat, CheckSize,
CheckWidth, CheckDefault,
Type, OwnHelp, HelpText,
OwnStat, StatText

[.ExitMacro](#) = *text*

[.Name](#) = *text*

[.Enabled](#) = True

[.OwnHelp](#) = True

[.HelpText](#) = *text*

[.OwnStatus](#) = True

[.StatusText](#) = *text*

End With

With myField.TextInput

[.Width](#) = *num*

[.Default](#) = *text*

[.EditType](#)

End With

With myField.CheckBox

[.Size](#) = *num*

[.AutoSize](#) = True

[.Default](#) = True

End With

InsertFrame

InsertIndex

InsertMergeField

InsertObject

Selection.Frames.[Add](#)

ActiveDocument.Indexes.[Add](#)

ActiveDocument.MailMerge.Fields.[Add](#)

ActiveDocument.Shapes.[AddOLEObject](#)

InsertPageBreak	Selection. InsertBreak Type:=wdPageBreak
InsertPageField	ActiveDocument.Fields. Add Range:=range, Ty
InsertPageNumbers	ActiveDocument.Sections(1).Footers(wdHeader
	Selection. InsertParagraphAfter
InsertPara	' or Selection. TypeParagraph
	ActiveDocument.Shapes. AddPicture
InsertPicture Name, LinkToFile, New	ActiveDocument.InlineShapes. New Range:=rar
InsertSectionBreak	Selection.Range. InsertBreak Type:=WdBreakTy
InsertSound	Selection.InlineShapes. AddOLEObject ClassTy
InsertSpike	NormalTemplate.AutoTextEntries("Spike"). Inse
InsertSubdocument	ActiveDocument.Subdocuments. AddFromFile I
InsertSymbol	Selection. InsertSymbol
InsertTableOfAuthorities	ActiveDocument.TablesOfAuthorities. Add
InsertTableOfContents	ActiveDocument.TablesOfContents. Add
InsertTableOfFigures	ActiveDocument.TablesOfFigures. Add
InsertTimeField	ActiveDocument.Fields. Add Range:=range, Ty
InsertWordArt	ActiveDocument.Shapes. AddOLEObject
InStr()	InStr()
Int()	Int()
	For Each xItem In AutoCorrect. FirstLetterExce
	If xItem. Name = "apt." Then isFound = True
	Next xItem
IsAutoCorrectException()	For Each aItem In AutoCorrect. TwoInitialCaps
	If aItem. Name = "THem" Then aExists = Tru
	Next aItem

For Each aProp In ActiveDocument.[CustomDo](#)

IsCustomDocumentProperty() If aProp.[Name](#) = "age" Then isFound = True

Next aProp

IsDocumentDirty() x = Not ActiveDocument.[Saved](#)

IsDocumentPropertyReadOnly() x = [WordBasic](#).IsDocumentPropertyReadOnly(i

IsExecuteOnly() x = ActiveDocument.VBProject.Protection

IsMacro() Not applicable in Word 2002

IsTemplateDirty() x = Not ActiveDocument.AttachedTemplate.[Sa](#)

Italic, Italic() Selection.Font.[Italic](#) = True



Visual Basic Equivalents J Through L

A B C D E F G H I J K L M N O P R S T U V W Y

J

JustifyPara, JustifyPara() Selection.Paragraphs.[Alignment](#) =
wdAlignParagraphJustify

K

KeyCode() x = KeyBindings(1).[KeyCode](#)

KeyMacro\$(x) =
KeyBindings(1).[Command](#)

Kill *filename* Kill *filename*

L

Language, Language\$(x) Selection.[LanguageID](#)
LCase()

LCase\$(x) ' or

LCase\$(x)

Left\$(x)

Left\$(x) ' or

Left(x)

LeftPara, LeftPara() Selection.Paragraphs.[Alignment](#) =
wdAlignParagraphLeft

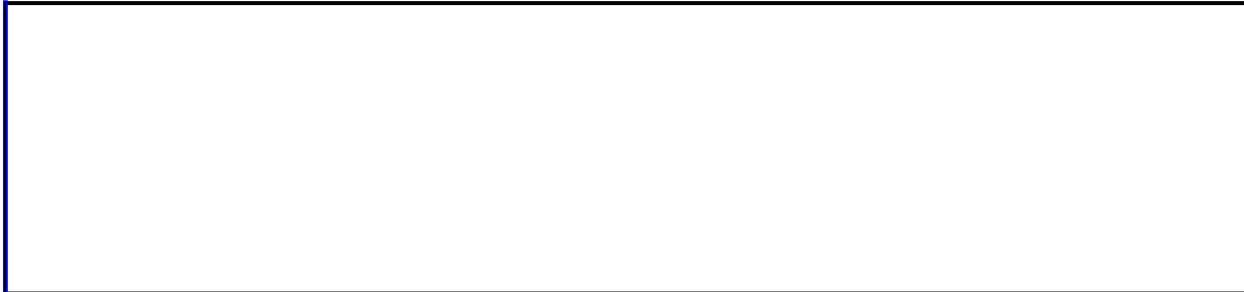
Len() Len()

Let Let

Line Input Line Input

Selection.[MoveDown](#) Unit:=wdLine, Count:=1,

LineDown, LineDown()	Extend:=wdMove
LineUp, LineUp()	Selection. MoveUp Unit:= wdLine, Count:=1, Extend:=wdMove
ListBox	ListBox Control
ListCommands	Application. ListCommands
LockDocument, LockDocument()	ActiveDocument.Subdocuments(1). Locked = True state = ActiveDocument.Subdocuments(1). Locked <i>' You can lock a single field or a group of fields within a range.</i>
LockFields	Selection.Fields. Locked = True ActiveDocument.Fields(1). Locked =True
Lof()	LOF()
LTrim\$()	LTrim()



Visual Basic Equivalents M

A B C D E F G H I J K L M N O P R S T U V W Y

M

MacID\$()	Macintosh only
MacroCopy	Application. OrganizerCopy
MacroDesc\$()	x = WordBasic . [MacroDesc\$](name)
MacroFileName\$()	Not applicable in Word 2002
MacroName\$()	x = WordBasic . [MacroName\$](num)
MacroNameFromWindow\$()	Not applicable in Word 2002
MacScript, MacScript\$()	Macintosh only ActiveWindow.View. Magnifier = True
Magnifier, Magnifier()	state = ActiveWindow.View. Magnifier
MailCheckNames	Application.MailMessage. CheckName
MailHideMessageHeader	Application.MailMessage. ToggleHeader With ActiveDocument.MailMerge .Check .Destination = <i>WdMailMergeDestination</i> .DataSource.FirstRecord = <i>num</i> .DataSource.LastRecord = <i>num</i> .SuppressBlankLines = True .MailSubject = <i>text</i> .MailAsAttachment = True
MailMerge <i>CheckErrors,</i> <i>Destination, MergeRecords, From,</i> <i>To, Suppression, MailMerge,</i> <i>MailSubject, MailAsAttachment,</i> <i>MailAddress</i>	

	.MailAddressFieldName = <i>text</i>
	.Execute
	End With
MailMergeAskToConvertChevrons, MailMergeAskToConvertChevrons()	FileConverters. ConvertMacWordChevrons <i>WdChevronConvertRule</i> state = FileConverters. ConvertMacWordCh
MailMergeCheck	ActiveDocument.MailMerge. Check FileConverters. ConvertMacWordChevrons <i>WdChevronConvertRule</i> state = FileConverters. ConvertMacWordCh
MailMergeConvertChevrons, MailMergeConvertChevrons()	ActiveDocument.MailMerge. ConvertMacWordChevrons <i>WdChevronConvertRule</i> state = FileConverters. ConvertMacWordCh
MailMergeCreateDataSource MailMergeCreateHeaderSource MailMergeDataForm	ActiveDocument.MailMerge. CreateDataSo Documents(<i>name</i>).MailMerge. CreateHeaderSource ActiveDocument. DataForm x = ActiveDocument.MailMerge.DataSource
MailMergeDataSource\$(0)	x =
MailMergeDataSource\$(1)	ActiveDocument.MailMerge.DataSource. H
MailMergeDataSource\$(2)	x = ActiveDocument.MailMerge.DataSource
MailMergeDataSource\$(3)	x = ActiveDocument.MailMerge.DataSource. H
MailMergeEditDataSource MailMergeEditHeaderSource MailMergeEditMainDocument MailMergeFindRecord MailMergeFirstRecord MailMergeFoundRecord()	Documents(<i>name</i>).MailMerge. EditDataSource Documents(1).MailMerge. EditHeaderSource ActiveDocument.MailMerge. EditMainDoc ActiveDocument.MailMerge.DataSource. F ActiveDocument.MailMerge.DataSource. A <i>wdFirstRecord</i> x = ActiveDocument.MailMerge.DataSource ActiveDocument.MailMerge.DataSource. A

MailMergeGotoRecord, MailMergeGotoRecord()	<i>num</i> x = ActiveDocument.MailMerge.DataSource
MailMergeHelper	Dialogs(wdDialogMailMergeHelper). Show
MailMergeInsertAsk	Documents(<i>name</i>).MailMerge.Fields. AddA
MailMergeInsertFillIn	Documents(<i>name</i>).MailMerge.Fields. AddF
MailMergeInsertIf	ActiveDocument.MailMerge.Fields. AddIf
MailMergeInsertMergeRec	ActiveDocument.MailMerge.Fields. AddMe
MailMergeInsertMergeSeq	ActiveDocument.MailMerge.Fields. AddMe
MailMergeInsertNext	Documents(1).MailMerge.Fields. AddNext
MailMergeInsertNextIf	ActiveDocument.MailMerge.Fields. AddNe
MailMergeInsertSet	ActiveDocument.MailMerge.Fields. AddSe
MailMergeInsertSkipIf	ActiveDocument.MailMerge.Fields. AddSk
MailMergeLastRecord	Documents(<i>name</i>).MailMerge.DataSource. wdLastRecord ActiveDocument.MailMerge. MainDocume <i>WdMailMergeMainDocType</i>
MailMergeMainDocumentType, MailMergeMainDocumentType()	state = ActiveDocument.MailMerge. MainL
MailMergeNextRecord	ActiveDocument.MailMerge.DataSource. A wdNextRecord
MailMergeOpenDataSource	Documents(1).MailMerge. OpenDataSource
MailMergeOpenHeaderSource	Documents(<i>name</i>).MailMerge. OpenHeader
MailMergePrevRecord	ActiveDocument.MailMerge.DataSource. A wdPreviousRecord
MailMergeQueryOptions	ActiveDocument.MailMerge.DataSource. Q
MailMergeReset	ActiveDocument.MailMerge. MainDocume wdNotAMergeDocument
MailMergeState()	theState = ActiveDocument.MailMerge. Sta
MailMergeToDoc	Documents(<i>name</i>).MailMerge. Destination : wdSendToNewDocument
MailMergeToPrinter	ActiveDocument.MailMerge. Destination =
MailMergeUseAddressBook	Not applicable in Word 2002

MailMergeViewData,
MailMergeViewData()

ActiveDocument.MailMerge.[ViewMailMerge](#)
True

x = ActiveDocument.MailMerge.[ViewMailMerge](#)

MailMessageDelete
MailMessageForward
MailMessageMove
MailMessageNext
MailMessagePrevious
MailMessageProperties
MailMessageReply
MailMessageReplyAll
MailSelectNames

Application.MailMessage.[Delete](#)
Application.MailMessage.[Forward](#)
Application.MailMessage.[DisplayMoveDialog](#)
Application.MailMessage.[GoToNext](#)
Application.MailMessage.[GoToPrevious](#)
Application.MailMessage.[DisplayProperties](#)
Application.MailMessage.[Reply](#)
Application.MailMessage.[ReplyAll](#)
Application.MailMessage.[DisplaySelectNames](#)
ActiveDocument.TablesOfAuthorities.[MailMergeViewData](#)

MarkCitation

ActiveDocument.TablesOfAuthorities.[MailMergeViewData](#)

MarkIndexEntry
MarkTableOfContentsEntry
MenuItemMacro\$(*name*)
MenuItemText\$(*name*)
MenuMode
MenuText\$(*name*)
MergeFieldName\$(*num*)
MergeSubdocument

ActiveDocument.Indexes.[MarkEntry](#)
ActiveDocument.TablesOfContents.[MarkEntry](#)
x = CommandBars(*name*).Controls(*num*).OnAction
x = CommandBars(*name*).Controls(*num*).Caption
[WordBasic](#).MenuMode
x = CommandBars.ActiveMenuBar.Controls(*num*).Caption
x =
ActiveDocument.MailMerge.DataSource.[FieldNames](#)
ActiveDocument.Subdocuments.[Merge](#)
[WordBasic](#).MicrosoftAccess

MicrosoftAccess

' or use the technique shown in Microsoft Excel
[WordBasic](#).MicrosoftExcel

' or

If Tasks.[Exists](#)("Microsoft Excel") = True Then

```

Tasks("Microsoft Excel").Activate

MicrosoftExcel      Tasks("Microsoft Excel").WindowState =
                    wdWindowStateMaximize

Else

                    Shell "C:\MSOffice\Excel\Excel.exe"

End If

WordBasic.MicrosoftFoxPro

MicrosoftFoxPro    ' or use the technique shown in Microsoft E

WordBasic.Mail

MicrosoftMail      ' or use the technique shown in Microsoft E

WordBasic.PowerPoint

MicrosoftPowerPoint ' or use the technique shown in Microsoft E

WordBasic.Project

MicrosoftProject   ' or use the technique shown in Microsoft E

WordBasic.Publisher

MicrosoftPublisher ' or use the technique shown in Microsoft E

WordBasic.Schedule

MicrosoftSchedule  ' or use the technique shown in Microsoft E

MicrosoftSystemInfo System.MSInfo
Mid$()

Mid$()             ' or

```

Minute()
MkDir *path_name*
Month()
MountVolume
MoveButton
MoveText

Mid()

Minute()
MkDir *path_name*
Month()
Application.[MountVolume](#)
CommandBars(*name*).Controls(1).Move
[WordBasic](#).MoveText
With CommandBars(*name*)

[.Top](#) = *num*

MoveToolbar

[.Left](#) = *num*

End With

CommandBars(*name*).Position = *MsoBarP*

MsgBox, MsgBox()

MsgBox, MsgBox()



Visual Basic Equivalents N

A B C D E F G H I J K L M N O P R S T U V W Y

N

Name	Name
NewToolbar	CommandBars. Add Selection. Move Unit:=wdCell, Count:=1
NextCell	' or Selection.Cells(1). Next .Select Selection. GoToNext What:=wdGoToField
NextField, NextField()	' or Selection. NextField
NextMisspelling	Selection. GoToNext What:=wdGoToSpellingError Selection. GoToNext What:=wdGoToObject
NextObject	' or Selection. MoveRight Unit:=wdItem Selection. GoToNext What:=wdGoToPage ' or
NextPage, NextPage()	ActiveWindow.View.Type = wdPrintView ActiveWindow. PageScroll Down:=1 x =

NextTab() ActiveDocument.Paragraphs(1).TabStops(1).[Next](#).Posit
 NextWindow ActiveWindow.Next.[Activate](#)
 NormalFontPosition Selection.Font.[Position](#) = 0
 NormalFontSpacing Selection.Font.[Spacing](#) = 0
 NormalStyle Selection.[Style](#) = wdStyleNormal
 With ActiveDocument.PageSetup
 .[DifferentFirstPageHeaderFooter](#) = True

 NormalViewHeaderArea .[OddAndEvenPagesHeaderFooter](#) = True
 Type, FirstPage,
 OddAndEvenPages,
 HeaderDistance,
 FooterDistance
 .[HeaderDistance](#) = num
 .[FooterDistance](#) = num

 End With

 ActiveWindow.View.[SeekView](#) = WdSeekView

 With ActiveDocument.Footnotes
 .[Location](#) = WdFootnoteLocation
 .[NumberingRule](#) = WdNumberingRule
 .[NumberStyle](#) = WdNoteNumberStyle
 .[StartingNumber](#) = num
 End With

 With ActiveDocument.Endnotes
 .[Location](#) = WdEndnoteLocation
 .[NumberingRule](#) = WdNumberingRule
 .[NumberStyle](#) = WdNoteNumberStyle

.StartingNumber = *num*

End With

Now()

Now



Visual Basic Equivalents O Through P

A B C D E F G H I J K L M N O P R S T U V W Y

O

OK	WordBasic.OK
OKButton	CommandButton control
On Error	<u>On Error</u>
OnTime	Application. OnTime
Open	Open
OpenSubdocument	ActiveDocument.Subdocuments(<i>name</i>). Open
OpenUpPara	Selection.Paragraphs. OpenUp
OptionButton	OptionButton control
OptionGroup	Frame control
	Application. OrganizerCopy
Organizer	Application. OrganizerDelete
	Application. OrganizerRename
OtherPane	ActiveWindow.ActivePane.Next. Activate
	Selection.Font. Outline = True
Outline, Outline()	x = Selection.Font. Outline
OutlineCollapse	ActiveWindow.View. CollapseOutline
OutlineDemote	Selection.Paragraphs. OutlineDemote
OutlineExpand	ActiveWindow.View. ExpandOutline
OutlineLevel()	aLevel = Selection.Paragraphs. OutlineLevel
	Selection.Range. Relocate

OutlineMoveDown Direction:=wdRelocateDown

OutlineMoveUp Selection.Range.[Relocate](#) Direction:=wdRelocateUp

OutlinePromote Selection.Paragraphs.[OutlinePromote](#)

 ActiveWindow.View.[ShowFirstLineOnly](#) = True

OutlineShowFirstLine,
OutlineShowFirstLine() x = ActiveWindow.View.[ShowFirstLineOnly](#)

OutlineShowFormat AcitveWindow.View.[ShowFormat](#) = True

Overtyping Options.[Overtyping](#) = True

P

PageDown, PageDown() Selection.[MoveDown](#) Unit:=wdScreen, Count:=1,
 Extend:=wdMove

PageUp, PageUp() Selection.[MoveUp](#) Unit:=wdScreen, Count:=1,
 Extend:=wdMove

ParaDown, ParaDown() Selection.[MoveDown](#) Unit:=wdParagraph, Count:=
 Extend:=wdMove

 ActiveDocument.Paragraphs(1).[KeepTogether](#) = Tr

ParaKeepLinesTogether,
ParaKeepLinesTogether() x = ActiveDocument.Paragraphs(1).[KeepTogether](#)

 ActiveDocument.Paragraphs(1).[KeepWithNext](#) = Tr

ParaKeepWithNext,
ParaKeepWithNext() x = ActiveDocument.Paragraphs(1).[KeepWithNext](#)

 ActiveDocument.Paragraphs(1).[PageBreakBefore](#) =

ParaPageBreakBefore,
ParaPageBreakBefore() x = ActiveDocument.Paragraphs(1).[PageBreakBefo](#)

ParaUp, ParaUp() Selection.[MoveUp](#) Unit:=wdParagraph, Count:=1,
 Extend:=wdMove

 ActiveDocument.Paragraphs(1).[WidowControl](#) = Tr

ParaWidowOrphanControl,
ParaWidowOrphanControl() x = ActiveDocument.Paragraphs(1).[WidowControl](#)

PasteButtonImage CommandBars(name).Controls(1).[PasteFace](#)

PasteFormat Selection.[PasteFormat](#)

PathFromMacPath\$()	x = WordBasic .PathFromMacPath\$(path)
PathFromWinPath\$()	x = WordBasic .PathFromWinPath\$(path)
PauseRecorder	WordBasic .PauseRecorder
Picture	Image control
	Selection. Move Unit:=wdCell, Count:=-1
	' or
PrevCell, PrevCell()	Selection.Cells(1). Previous .Select
	Selection. GoToPrevious What:=wdGoToField
	' or
PrevField, PrevField()	Selection. PreviousField
	Selection. GoToPrevious What:=wdGoToObject
	' or
PrevObject	Selection. MoveLeft Unit:=wdItem
	Selection. GoToPrevious What:=wdGoToPage
	' or
PrevPage, PrevPage()	ActiveWindow.View.Type = wdPrintView
	ActiveWindow. PageScroll Up:=1
	x =
PrevTab()	ActiveDocument.Paragraphs(1).TabStops(1). Previo
PrevWindow	ActiveWindow.Previous. Activate
Print	Print
PromoteList	Selection.Range.ListFormat. ListIndent
PushButton	CommandButton control
PutFieldData	ActiveDocument.Fields(1). Data = text



Visual Basic Equivalents R

A B C D E F G H I J K L M N O P R S T U V W Y

R

Read	Input #
Redim	ReDim
REM	REM
RemoveAllDropDownItems	ActiveDocument.FormFields(1).DropDown.ListEnt
RemoveBulletsNumbers	Selection.Range.ListFormat. RemoveNumbers
RemoveDropDownItem	ActiveDocument.FormFields(1).DropDown.ListEnt While Selection.Frames.Count > 0
RemoveFrames	Selection.Frames(1). Delete
	Wend
RemoveSubdocument	ActiveDocument.Subdocuments(1). Delete
RenameMenu	CommandBars.ActiveMenuBar.Controls(<i>name</i>).Caption <i>newname</i>
RepeatFind	Application. Run MacroName:="RepeatFind"
ResetButtonImage	CommandBars(<i>name</i>).Controls(1).Reset
ResetChar, ResetChar()	Selection.Font. Reset ActiveDocument.Endnotes. ResetContinuationNotice ActiveDocument.Footnotes. ResetContinuationNotice
ResetNoteSepOrNotice	ActiveDocument.Endnotes. ResetContinuationSeparator ActiveDocument.Footnotes. ResetContinuationSeparator
ResetPara, ResetPara()	Selection.Paragraphs.Reset Right()

Right\$()	' or Right\$()
RightPara, RightPara() Rmdir <i>path</i> Rnd(<i>number</i>)	Selection.Paragraphs. Alignment = wdAlignParagraf Rmdir <i>path</i> Rnd(<i>number</i>) RTrim()
RTrim\$()	' or RTrim\$()
RunPrintManager	Not available



Visual Basic Equivalents S

A B C D E F G H I J K L M N O P R S T U V W Y

S

	ActiveDocument.AttachedTemplate. Save
SaveTemplate	' or Templates(<i>name</i>). Save
ScreenRefresh	Application. ScreenRefresh
ScreenUpdating, ScreenUpdating()	Application. ScreenUpdating = True
Second(<i>time</i>)	Second(<i>time</i>)
Seek <i>filenumber</i> , position	Seek[#] <i>filenumber</i> , <i>position</i>
Seek(<i>filenumber</i>)	Seek(<i>filenumber</i>)
Select Case	Select Case
SelectCurAlignment	Selection. SelectCurrentAlignment
SelectCurColor	Selection. SelectCurrentColor
SelectCurFont	Selection. SelectCurrentFont
SelectCurIndent	Selection. SelectCurrentIndent
SelectCurSentence	Selection.Sentences(1). Select
SelectCurSpacing	Selection. SelectCurrentSpacing
SelectCurTabs	Selection. SelectCurrentTabs
SelectCurWord	Selection.Words(1). Select
Selection\$()	<i>text</i> = Selection. Text
SelectionFileName\$()	<i>aFileName</i> = Selection.Document. FullName
SelInfo(<i>Type</i>)	<i>x</i> = Selection. Information (<i>WdInformation</i>)
SelType()	<i>aType</i> = Selection. Type (<i>Type</i>)

SelType 1	Selection. Collapse Direction:=wdCollapseStart
SendKeys <i>keys, wait</i>	SendKeys <i>keys, wait</i>
SentLeft 1,1	Selection.Sentences(1). Previous (Unit:=wdSentence, Count:=1).Select
SentRight 1, 1	Selection.Sentences(1). Next (Unit:=wdSentence, Count:=1).Select
SetAttr <i>filename, attribute</i>	SetAttr <i>filename, attribute</i>
SetAutoText	Templates(<i>name</i>).AutoTextEntries. Add
SetDocumentDirty 1	ActiveDocument. Saved = False ActiveDocument.BuiltInDocumentProperties.Add
SetDocumentProperty	' or ActiveDocument.CustomDocumentProperties.Add
SetDocumentPropertyLink <i>name, source</i>	ActiveDocument.CustomDocumentProperties(<i>name</i>). = <i>source</i>
SetDocumentVar <i>name, value</i>	ActiveDocument.Variables. Add <i>name, value</i>
SetEndOfBookmark <i>name</i>	<i>range</i> .Bookmarks(<i>name</i>). Start = <i>range</i> .Bookmarks(<i>n</i>
SetEndOfBookmark <i>name1, name2</i>	ActiveDocument.Bookmarks(<i>name1</i>).End = ActiveDocument.Bookmarks(<i>name2</i>).End
SetFileCreatorAndType	Macintosh only
SetFormResult <i>name, "text"</i>	ActiveDocument.FormFields(<i>name</i>). Result = "text"
SetFormResult <i>name, 1</i>	ActiveDocument.FormFields(<i>name</i>).CheckBox. Value
SetFormResult <i>name, num</i>	ActiveDocument.FormFields(<i>name</i>).DropDown. Value
SetFormResult <i>name, , default</i>	Use the Default property with a CheckBox, DropDown TextInput object.
SetPrivateProfileString	

<i>section, key, setting, filename</i>	System. PrivateProfileString (filename, section, key) =
SetProfileString <i>section, key, setting</i>	System. ProfileString (section, key) = setting
SetSelRange <i>charpos1, charpos2</i>	ActiveDocument.Range(Start:=charpos1, End:=charpos2).Bookmarks(name).End = range.Bookmarks(name).End
SetStartOfBookmark <i>name</i>	ActiveDocument.Bookmarks(book1).Start = ActiveDocument.Bookmarks(book2).Start
SetStartOfBookmark <i>book1, book2</i>	Documents(name).AttachedTemplate.Saved = True
SetTemplateDirty 0	' or Templates(name).Saved = True
Sgn()	Sgn()
ShadingPattern, ShadingPattern()	Selection.Shading.Texture = WdTextureIndex
Shadow, Shadow()	Selection.Font.Shadow = True x = Selection.Font.Shadow
Shell	Shell Windows(1).View.ShowAll = True
ShowAll, ShowAll()	x = ActiveWindow.View.ShowAll
ShowAllHeadings	ActiveWindow.View.ShowAllHeadings
ShowAnnotationBy <i>name</i>	ActiveDocument.Comments.ShowBy = name
ShowClipboard	Application.ShowClipboard
ShowHeadingNumber	Windows(name).View.ShowHeading Level:=num
ShowMe	Application.ShowMe
ShowNextHeaderFooter	ActiveWindow.View.NextHeaderFooter
ShowPrevHeaderFooter	ActiveWindow.View.PreviousHeaderFooter

ShowVars	Add a watch expression in the Visual Basic Editor
ShrinkFont	Selection.Font. Shrink
ShrinkFontOnePoint	Selection.Font.Size = Selection.Font. Size - 1
ShrinkSelection	Selection. Shrink
SizeToolbar <i>name, width</i>	CommandBars(<i>name</i>).Width = <i>num</i>
SkipNumbering, SkipNumbering()	Selection.Range.ListFormat. RemoveNumbers
SmallCaps, SmallCaps()	Selection.Font. SmallCaps = True
SortArray	WordBasic .SortArray Selection.Paragraphs. Space1
SpacePara1, SpacePara1()	x = Selection.Paragraphs. LineSpacing
	Selection.Paragraphs. Space15
SpacePara15, SpacePara15()	x = Selection.Paragraphs. LineSpacing
	Selection.Paragraphs. Space2
SpacePara2, SpacePara2()	x = Selection.Paragraphs. LineSpacing
	ActiveDocument.Content. SpellingChecked = True
SpellChecked, SpellChecked()	x = ActiveDocument.Content. SpellingChecked
Spike	NormalTemplate.AutoTextEntries. AppendToSpike
SplitSubdocument	ActiveDocument.Subdocuments(1). Split Range:=ran
StartOfColumn, StartOfColumn()	Selection. StartOf Unit:=wdColumn, Extend:=wdMov
StartOfDocument, StartOfDocument()	Selection. HomeKey Unit:=wdStory, Extend:=wdMov
StartOfLine, StartOfLine()	Selection. HomeKey Unit:=wdLine, Extend:=wdMov
StartOfRow, StartOfRow()	Selection. StartOf Unit:=wdRow, Extend:=wdMove
StartOfWindow, StartOfWindow()	Selection. MoveUp Unit:=wdWindow
Stop	Stop

Str\$(<i>number</i>)	Str(<i>number</i>)
	Str\$(<i>number</i>)
Strikethrough, Strikethrough()	Selection.Font. StrikeThrough = True
	String(<i>count</i> , <i>character</i>)
String\$(<i>count</i> , <i>character</i>)	String\$(<i>count</i> , <i>character</i>)
Style	Selection. Style = wdStyleHeading1
StyleDesc\$()	x = Selection.Style. Description
StyleName\$()	x = Selection.Style. NameLocal
Sub...End Sub	Sub...End Sub
	Selection.Font. Subscript = True
Subscript, Subscript()	x = Selection.Font. Subscript
	Selection.Font. Superscript = True
Superscript, Superscript()	x = Selection.Font. Superscript
SymbolFont	Selection.Font. Name = "Symbol"

Visual Basic Equivalents T

A B C D E F G H I J K L M N O P R S T U V W Y

T

TabLeader\$(<i>pos</i>)	aType = Selection.Paragraphs(<i>num</i>).TabStops(
TableAutoFormat	ActiveDocument.Tables(1). AutoFormat
TableAutoSum	ActiveDocument.Tables(1).Cell(<i>row</i> , <i>column</i>).
TableColumnWidth <i>ColumnWidth, RulerStyle</i>	ActiveDocument.Tables(1).Columns. SetWidth Rulerstyle:= <i>wdRulerStyle</i>
TableColumnWidth <i>AutoFit</i>	ActiveDocument.Tables(1).Columns. AutoFit
TableColumnWidth <i>NextColumn</i>	Selection.Columns(1). Next .Select
TableColumnWidth <i>PrevColumn</i>	Selection.Columns(1). Previous .Select
TableColumnWidth <i>SpaceBetweenCols</i>	ActiveDocument.Tables(1).Rows. SpaceBetween
TableDeleteCells <i>ShiftCells</i>	ActiveDocument.Tables(1).Cell(<i>row</i> , <i>column</i>). ActiveDocument.Tables(1).Columns(<i>num</i>). Delete
TableDeleteColumn	' or ActiveDocument.Tables(1).Columns. Delete ActiveDocument.Tables(1).Rows(<i>num</i>). Delete
TableDeleteRow	' or ActiveDocument.Tables(1).Rows. Delete
TableFormula	ActiveDocument.Tables(1).Cell(<i>row</i> , <i>column</i>).

TableGridlines, TableGridlines() ActiveWindow.View.[TableGridlines](#) = True
 x = ActiveWindow.View.[TableGridlines](#)
 Selection.Tables(1).Rows(num).[HeadingFormat](#)
 ' or
 TableHeadings, TableHeadings() Selection.Tables(1).Rows.[HeadingFormat](#) = Tr
 x = Selection.Tables(1).Rows.[HeadingFormat](#)
 TableInsertCells Selection.Tables(1).Columns(num).Cells.[Add](#)
 TableInsertColumn Selection.Tables(1).Columns.[Add](#)
 TableInsertRow Selection.Tables(1).Rows.[Add](#)
 TableInsertTable *NumColumns, NumRows* ActiveDocument.Tables.[Add](#) Range:=range, N
 TableInsertTable *NumColumns, NumRows, Format, Apply* ActiveDocument.Tables.Add(Range:=range, N
 NumColumns:=num).[AutoFormat](#)
 TableInsertTable *NumColumns, NumRows, ConvertFrom* Selection.[ConvertToTable](#) Separator:=WdTable
 NumColumns:=num
 TableMergeCells Selection.Cells.[Merge](#)
 With ActiveDocument.Tables(num).Rows(num
 .[SetHeight](#) RowHeight:= num, HeightRule:=
 TableRowHeight *RulerStyle, LineSpacingRule, LineSpacing, LeftIndent, Alignment, AllowRowSplit* .[Alignment](#) = WdRowAlignment
 .[SetLeftIndent](#) LeftIndent:=num, RulerStyle:
 TableRowHeight *NextColumn* .[AllowBreakAcrossPages](#) = True
 TableRowHeight *PrevColumn* End With
 Selection.Rows(1).[Next](#).Select
 Selection.Rows(1).[Previous](#).Select

TableSelectColumn	Selection.Tables(1).Columns(num). Select
TableSelectRow	Selection.Tables(1).Rows(num). Select
TableSelectTable	ActiveDocument.Tables(1). Select
TableSort	ActiveDocument.Tables(1). Sort
TableSortAToZ	ActiveDocument.Tables(1). SortAscending
TableSortZToA	ActiveDocument.Tables(1). SortDescending
TableSplit	Selection.Tables(1). Split
TableSplitCells	Selection.Tables(1).Cells(row, column). Split
TableToText	Selection.Tables(1). ConvertToText
TableUpdateAutoFormat	Selection.Tables(1). UpdateAutoFormat
TabType()	x = Selection.ParagraphFormat.TabStops(1).A
Text	Label control
TextBox	TextBox control
TextFormField	ActiveDocument.FormFields. Add Range:=ran
TextToTable]	Selection. ConvertToTable
	Time()
Time\$()	' or
	Time\$()
TimeSerial()	TimeSerial
TimeValue()	TimeValue
	No direct equivalent
TipWizard	' displays a special tip when Word is launched
	Assistant.FeatureTips = True
	Dim x As Long
Today()	x = DateSerial(Year(Date), Month(Date), Day(
ToggleFieldDisplay	Selection.Fields. ToggleShowCodes
ToggleFull	ActiveWindow.View. FullScreen = Not Active\

ToggleHeaderFooterLink	ActiveDocument.Sections(2).Headers(wdHead
ToggleMainTextLayer	ActiveWindow.View. ShowMainTextLayer = N
TogglePortrait	ActiveWindow.View.ShowMainTextLayer
ToggleScribbleMode	ActiveDocument.PageSetup. TogglePortrait
ToolbarButtonMacro\$(<i>name</i> = ToolbarName\$(ToolbarState(<i>name</i>))	WordBasic .ToggleScribbleMode x = CommandBars(<i>name</i>).Controls(1).OnActic <i>name</i> = CommandBars(<i>num</i>). Name CommandBars(<i>name</i>). Visible = True WordBasic .ToolsAddRecordDefault ' or if the data source is a Word table
ToolsAddRecordDefault	Selection.Tables(1).Cell(Row:=Selection.Infor – Column:=Selection.Information(wdMaximu Selection.MoveRight Unit:=wdCell
ToolsAdvancedSettings	Not available with any 32-bit version of Windc With AutoCorrect .CorrectInitialCaps = True
ToolsAutoCorrect <i>InitialCaps, SentenceCaps, Days, CapsLock, ReplaceText</i>	.CorrectSentenceCaps = True
ToolsAutoCorrect <i>SmartQuotes</i>	.CorrectDays = True
ToolsAutoCorrect <i>Formatting, Replace, With, Add</i>	.CorrectCapsLock = True
ToolsAutoCorrect <i>Formatting, Replace, With, Add</i>	.ReplaceText = True
ToolsAutoCorrect .Replace = <i>text</i> , .Delete	End with Options. AutoFormatAsYouTypeReplaceQuote AutoCorrectEntries. AddRichText Name:= <i>text</i> AutoCorrectEntries. Add Name:= <i>text</i> , Value:=

AutoCorrectEntries(*name*).[Delete](#)

ToolsAutoCorrectCapsLockOff,
ToolsAutoCorrectCapsLockOff() AutoCorrect.[CorrectCapsLock](#) = True

ToolsAutoCorrectDays,
ToolsAutoCorrectDays() AutoCorrect.[CorrectDays](#) = True

ToolsAutoCorrectExceptions *Tab*
= 0, *Name*, *Add*

ToolsAutoCorrectExceptions *Tab* FirstLetterExceptions.[Add](#) *name*
= 1, *Name*, *Add*

ToolsAutoCorrectExceptions *Tab* TwoInitialCapsExceptions.[Add](#) *name*
= 0, *AutoAdd* AutoCorrect.[FirstLetterAutoAdd](#) = True

ToolsAutoCorrectExceptions *Tab* AutoCorrect.[TwoInitialCapsAutoAdd](#) = True
= 1, *AutoAdd*

ToolsAutoCorrectExceptions *Tab* FirstLetterExceptions(*name*).[Delete](#)
= 0, *Name*, *Delete* TwoInitialCapsExceptions(*name*).[Delete](#)

ToolsAutoCorrectExceptions *Tab*
= 1, *Name*, *Delete*

ToolsAutoCorrectInitialCaps,
ToolsAutoCorrectInitialCaps() AutoCorrect.[CorrectInitialCaps](#) = True

ToolsAutoCorrectReplaceText,
ToolsAutoCorrectReplaceText() AutoCorrect.[ReplaceText](#) = True

ToolsAutoCorrectSentenceCaps,
ToolsAutoCorrectSentenceCaps() AutoCorrect.[CorrectSentenceCaps](#) = True

ToolsAutoCorrectSmartQuotes,
ToolsAutoCorrectSmartQuotes() Options.[AutoFormatAsYouTypeReplaceQuote](#)

ToolsBulletListDefault Selection.Range.ListFormat.[ApplyBulletDefau](#)
With [ListGalleries](#)(wdNumberGallery).[ListTer](#)

[.NumberFormat](#) = "%1."

[.TrailingCharacter](#) = *WdTrailingCharacter*

ToolsBulletsNumbers *Replace*,
Font, *CharNum*, *Type*,
FormatOutline, *AutoUpdate*,
FormatNumber, *Punctuation*,
StartAt, *Points*, *Hang*, *Indent*,
Remove

[.NumberStyle](#) = *WdListNumberStyle*

[.Alignment](#) = *WdListLevelAlignment*

[.TextPosition](#) = *InchesToPoints(num)*

[.TabPosition](#) = *InchesToPoints(num)*

[.ResetOnHigher](#) = *True*

[.StartAt](#) = *num*

[.Font](#).Size = *num*

End With

Selection.Range.ListFormat.[ApplyListTemplat](#)

ListTemplate:=ListGalleries(wdNumberG

ToolsCalculate, ToolsCalculate()

Selection.Range.[Calculate](#)

ToolsCompareVersions

ActiveDocument.[Compare](#)

ToolsCreateEnvelope

PrintEnvLabel

ActiveDocument.Envelope.[Insert](#)

ToolsCreateEnvelope

AddToDocument

ActiveDocument.Envelope.[PrintOut](#)

ToolsCreateLabels

PrintEnvLabel

Application.MailingLabel.[PrintOut](#)

ToolsCreateLabels

AddToDocument

Application.MailingLabel.[CreateNewDocume](#)

With Dialogs(*WdWordDialog*)

[.DefaultTab](#) = *WdWordDialogTab*

ToolsCustomize *Tab*

[.Show](#)

End With

[CustomizationContext](#) = *template or document*

KeyBindings.[Add](#)

ToolsCustomizeKeyboard
*KeyCode, KeyCode2, Category,
Name, Add, Remove, ResetAll,
CommandValue, Context*

[CustomizationContext](#) = *template or document*

FindKey(BuildKeyCode(Wdkey, Wdkey)).[Disa](#)

[CustomizationContext](#) = *template or document*

KeyBindings.[ClearAll](#)

[CustomizationContext](#) = *template or document*

ToolsCustomizeMenuBar
*Context, Position, MenuType,
MenuText, Menu, Add, Remove,
Rename*

CommandBars(*name*).[Delete](#)

CommandBars.[Add](#)

CommandBars(*name*).[Name](#) = *text*

[CustomizationContext](#) = *template or document*

ToolsCustomizeMenus
*MenuType, Position, Category,
Name, Menu, AddBelow,
MenuText, Rename, Add,
Remove, ResetAll,
CommandValue, Context*

CommandBars(*name*).Controls(*num*).[Delete](#)

CommandBars(*name*).Controls.[Add](#) Type:=ms
Before:=*num*

CommandBars(*name*).Controls(*num*).Caption :

ToolsGetSpelling,
ToolsGetSpelling()

[GetSpellingSuggestions](#)

ToolsGetSynonyms,
ToolsGetSynonyms()

[SynonymInfo](#)

ToolsGrammar

ActiveDocument.[CheckGrammar](#)

' *enumerate the ReadabilityStatistics collection*

i = 1

ToolsGrammarStatisticsArray

For Each aStat In ActiveDocument.[Readability](#)

aArray(i) = aStat.[Value](#)

i = i + 1

Next aStat

With ActiveDocument

.[AutoHyphenation](#) = True

.[HyphenateCaps](#) = True

.[HyphenationZone](#) = num

.[ConsecutiveHyphensLimit](#) = num

End With

ToolsHyphenation

AutoHyphenation,

HyphenateCaps,

HyphenationZone,

LimitConsecutiveHyphens

ActiveDocument.[ManualHyphenation](#)

Selection.Range.[LanguageID](#) = *WdLanguageI*

ActiveDocument.Styles(wdStyleNormal).Lang

Application.[Run](#)

Application.[OrganizerDelete](#)

Application.[OrganizerRename](#)

With [Dialogs](#)(wdDialogToolsMacro)

.Show = "templateName"

.Name = "macroName"

.Edit = True

.Execute

ToolsHyphenationManual

ToolsLanguage *Language,*

Default

ToolsMacro *Name, Run, Edit,*

*Show, Delete, Rename,
Description, NewName, SetDesc*

End With

With [Dialogs](#)(wdDialogToolsMacro)

.Show = "templateName"

.Name = "macroName"

.Description = "newDescription"

.SetDesc = True

.Execute

End With

ToolsManageFields

ToolsMergeRevisions

ToolsNumberListDefault

ToolsOptions

Application.[Run](#) MacroName:="ToolsManage

ActiveDocument.[Merge](#) FileName:=*name*

Selection.Range.ListFormat.[ApplyNumberDef](#)

Dialogs(*WdWordDialog*).[Show](#)

With Options

.[AutoFormatPreserveStyles](#) = True

.[AutoFormatApplyHeadings](#) = True

.[AutoFormatApplyLists](#) = True

.[AutoFormatApplyOtherParas](#) = True

.[AutoFormatReplaceQuotes](#) = True

.[AutoFormatReplaceSymbols](#) = True

.[AutoFormatApplyBulletedLists](#) = True

.[AutoFormatReplaceOrdinals](#) = True

ToolsOptionsAutoFormat

PreserveStyles,

ApplyStylesHeadings,

ApplyStylesLists,

ApplyStylesOtherParas,

ReplaceQuotes, ReplaceSymbols,

ApplyBulletedLists,

ReplaceOrdinals,

ReplaceFractions,

ShowOptionsFor

ToolsOptionsAutoFormat

ApplyBorders,

.[AutoFormatReplaceFractions](#) = True

ApplyBulletedLists,
ApplyStylesHeadings,
ApplyNumberedLists,
ReplaceFractions,
ReplaceOrdinals,
ReplaceQuotes, ReplaceSymbols,
ShowOptionFor

There is no Visual Basic
equivalent for the following
arguments: *AdjustParaMarks,*
AdjustTabsSpaces,
ReplaceBullets,
AdjustEmptyParas.

End With

With Options

[.AutoFormatAsYouTypeApplyBorders](#) = True

[.AutoFormatAsYouTypeApplyBulletedLists](#)

[.AutoFormatAsYouTypeApplyHeadings](#) = True

[.AutoFormatAsYouTypeApplyNumberedLists](#)

[.AutoFormatAsYouTypeReplaceFractions](#) =

[.AutoFormatAsYouTypeReplaceOrdinals](#) =

[.AutoFormatAsYouTypeReplaceQuotes](#) = True

[.AutoFormatAsYouTypeReplaceSymbols](#) =

End With

ToolsOptionsCompatibility

ActiveDocument.[Compatibility](#) Type:=WdCon

With Options

[.ReplaceSelection](#) = True

[.AllowDragAndDrop](#) = True

[.AutoWordSelection](#) = True

[.INSKeyForPaste](#) = True

[.Overtyping](#) = True

[.SmartCutPaste](#) = True

[.AllowAccentedUppercase](#) = True

[.PictureEditor](#) = *text*

ToolsOptionsEdit
ReplaceSelection, DragAndDrop,
AutoWordSelection, InsForPaste,
Overtyping, SmartCutPaste,
AllowAccentedUppercase,
PictureEditor, TabIndent

[.TabIndentKey](#) = True

End With

ToolsOptionsFileLocations

Options.[DefaultFilePath](#)(*WdDefaultFilePath*) :

With Options

[.Pagination](#) = True

[.WPHelp](#) = True

[.WPDocNavKeys](#) = True

[.BlueScreen](#) = True

[.EnableSound](#) = True

[.UpdateLinksAtOpen](#) = True

ToolsOptionsGeneral *Pagination,*
WPHelp, WPDocNavKeys,
BlueScreen, ErrorBeeps,
UpdateLinks, SendMailAttach,
Units, ButtonFieldClicks,
ShortMenuNames,
RTFInClipboard,
ConfirmConversions,
TipWizardActive, RecentFiles,
RecentFileCount

[.SendMailAttach](#) = True

[.MeasurementUnit](#) = *WdUnits*

[.ButtonFieldClicks](#) = *num*

[.ShortMenuNames](#) = True

[.RTFInClipboard](#) = True

[.ConfirmConversions](#) = True

End With

Assistant.[ActivateWizard](#)

With Application

[.DisplayRecentFiles](#) = True

.RecentFiles.[Maximum](#) = *num*

End With

With Options

[.CheckGrammarWithSpelling](#) = True

ToolsOptionsGrammar *Options*,
CheckSpelling, *ShowStatistics*

[.ShowReadabilityStatistics](#) = True

End With

ActiveDocument.[ActiveWritingStyle](#)(*language*

With Options

[.PrintDraft](#) = True

[.PrintReverse](#) = True

[.UpdateFieldsAtPrint](#) = True

[.PrintProperties](#) = True

[.PrintFieldCodes](#) = True

[.PrintComments](#) = True

[.PrintHiddenText](#) = True

ToolsOptionsPrint *Draft*,
Reverse, *UpdateFields*,
Summary, *ShowCodes*,
Annotations, *ShowHidden*,
EnvFeederInstalled,
UpdateLinks, *Background*,
DrawingObjects, *DefaultTray*,
FormsData, *FractionalWidths*,
PSOverText

[.EnvelopeFeederInstalled](#) = True

[.UpdateLinksAtPrint](#) = True

[.PrintBackground](#) = True

[.PrintDrawingObjects](#) = True

[.DefaultTray](#) = *text*

[.DefaultTrayID](#) = *WdPaperTray*

End With

With ActiveDocument

[.PrintFormsData](#) = True

[.PrintFractionalWidths](#) = True

[.PrintPostScriptOverText](#) = True

End With

With Options

[.InsertedTextMark](#) = *WdInsertedTextMark*

[.DeletedTextMark](#) = *WdDeletedTextMark*

[.RevisedLinesMark](#) = *WdRevisedLinesMark*

[.InsertedTextColor](#) = *WdColorIndex*

[.DeletedTextColor](#) = *WdColorIndex*

[.RevisedLinesColor](#) = *WdColorIndex*

[.DefaultHighlightColorIndex](#) = *WdColorInd*

End With

With Options

[.CreateBackup](#) = True

[.AllowFastSave](#) = True

[.SavePropertiesPrompt](#) = True

[.SaveNormalPrompt](#) = True

[.BackgroundSave](#) = True

ToolsOptionsRevisions

InsertedTextMark,

DeletedTextMark,

RevisedLinesMark,

InsertedTextColor,

DeletedTextColor,

RevisedLinesColor,

HighlightColor

ToolsOptionsSave *CreateBackup,*

FastSaves, SummaryPrompt,

GlobalDotPrompt,

*NativePictureFormat, AutoSave,
SaveInterval*

[.SaveInterval](#) = number

End With

*ToolsOptionsSave FormsData,
Password, WritePassword,
RecommendReadOnly,
EmbedFonts*

With ActiveDocument

[.SaveFormsData](#) = True

[.Password](#) = text

[.WritePassword](#) = text

[.ReadOnlyRecommended](#) = True

[.EmbedTrueTypeFonts](#) = True

End With

With Options

[.SuggestSpellingCorrections](#) = True

[.SuggestFromMainDictionaryOnly](#) = True

[.IgnoreUppercase](#) = True

[.IgnoreMixedDigits](#) = True

[.CheckSpellingAsYouType](#) = True

*ToolsOptionsSpelling
AlwaysSuggest,
SuggestFromMainDictOnly,
IgnoreAllCaps,
IgnoreMixedDigits,
ResetIgnoreAll, Type,
CustomDictn,
AutomaticSpellChecking,
HideSpellingErrors,
RecheckDocument*

End With

With ActiveDocument

[.SpellingChecked](#) = False

[.ShowSpellingErrors](#) = True

End With

Application.[ResetIgnoreAll](#)

Languages(*wdLanguageID*).[SpellingDictionary](#)

CustomDictionaries.[Add](#)

With Application

[.UserName](#) = *text*

ToolsOptionsUserInfo *Name,*
Initials, Address

[.UserInitials](#) = *text*

[.UserAddress](#) = *text*

End With

With ActiveWindow.View

[.Draft](#) = True

[.WrapToWindow](#) = True

[.ShowPicturePlaceHolders](#) = True

[.ShowFieldCodes](#) = True

[.ShowBookmarks](#) = True

[.FieldShading](#) = *WdFieldShading*

.Parent.[.DisplayHorizontalScrollBar](#) = True

.Parent.[.DisplayVerticalScrollBar](#) = True

.Parent.[.StyleAreaWidth](#) = *num*

[.ShowTabs](#) = True

[.ShowSpaces](#) = True

[.ShowParagraphs](#) = True

[.ShowHyphens](#) = True

ToolsOptionsView *DraftFont,*
WrapToWindow,
PicturePlaceHolders,
FieldCodes, BookMarks,
FieldShading, Hscroll, Vscroll,
StyleAreaWidth, Tabs, Spaces,
Paras, Hyphens, Hidden,
ShowAll, Drawings, Anchors,
TextBoundaries, Vruler,
Highlight

ToolsOptionsView *StatusBar* .[ShowHiddenText](#) = True
. [ShowAll](#) = True
. [ShowDrawings](#) = True
. [ShowObjectAnchors](#) = True
. [ShowTextBoundaries](#) = True
.Parent.[DisplayVerticalRuler](#) = True
. [ShowHighlight](#) = True

End With

Application.[DisplayStatusBar](#) = True

ToolsProtectDocument ActiveDocument.[Protect](#)
ToolsProtectSection *Protect*, ActiveDocument.Sections(*num*).[ProtectedForF](#)
Section [WordBasic](#).ToolsRemoveRecordDefault

ToolsRemoveRecordDefault ' *or if the data source is a Word table*
Selection.Tables(1).Rows(1).[Delete](#)

ToolsRepaginate ActiveDocument.[Repaginate](#)
ActiveDocument.[ShowRevisions](#) = True

ToolsReviewRevisions Selection.[NextRevision](#)
ShowMarks, HideMarks, Wrap, Selection.[PreviousRevision](#)
FindPrevious, FindNext,
AcceptRevisions, RejectRevisions Selection.Range.Revisions.[AcceptAll](#)

Selection.Range.Revisions.[RejectAll](#)

ToolsRevisionAuthor\$(*)* anAuthor = Selection.Range.Revisions(1).[Autl](#)

ToolsRevisionDate\$()	aDate = ActiveDocument.Revisions(1). Date With ActiveDocument .TrackRevisions = True .PrintRevisions = True .ShowRevisions = True
ToolsRevisions <i>MarkRevisions, ViewRevisions, PrintRevisions, AcceptAll, RejectAll</i>	End With With Selection.Range.Revisions .AcceptAll .RejectAll End With
ToolsRevisionType() ToolsShrinkToFit ToolsSpelling ToolsSpellingRecheckDocument ToolsSpellSelection ToolsThesaurus ToolsUnprotectDocument ToolsWordCount <i>CountFootnotes, Pages, Words, Characters, Paragraphs, Lines</i>	aType = ActiveDocument.Revisions(1). Type ActiveDocument. FitToPages ActiveDocument. CheckSpelling ActiveDocument. SpellingChecked = False Selection.Range. CheckSpelling Selection.Range. CheckSynonyms ActiveDocument. UnProtect (Password:= <i>text</i>) ActiveDocument. ComputeStatistics Statistic:= IncludeFootnotesAndEndnotes:=True

Visual Basic Equivalents U Through V

A B C D E F G H I J K L M N O P R S T U V W Y

U

	<code>UCase(string)</code>
<code>UCase\$(string)</code>	' or <code>UCase\$(string)</code>
<code>Underline,</code> <code>Underline()</code>	<code>ActiveDocument.Words(1).Underline = True</code> <code>status = Selection.Font.Underline</code>
<code>UnHang</code>	<code>ActiveDocument.Paragraphs(1).TabHangingIndent</code> <code>Count:=-1</code>
<code>UnIndent</code>	<code>Selection.Paragraphs.TabIndent Count:=-1</code> <code>Selection.Range.Fields.Unlink</code>
<code>UnlinkFields</code>	<code>ActiveDocument.Fields(num).Unlink</code>
<code>UnlockFields</code>	<code>Selection.Paragraphs(1).Range.Fields.Locked = False</code> <code>ActiveDocument.Fields(num).Locked = False</code>
<code>UpdateFields</code>	<code>ActiveDocument.Sections(1).Range.Fields.Update</code> <code>ActiveDocument.Fields(num).Update</code>
<code>UpdateSource</code>	<code>ActiveDocument.Paragraphs(1).Range.Fields.UpdateSource</code> <code>ActiveDocument.Fields(num).UpdateSource</code>

V

Val(<i>text</i>)	Val(<i>text</i>)
ViewAnnotations	ActiveWindow.View. SplitSpecial = wdPaneComments
ViewBorderToolbar	CommandBars("Borders"). Visible = True ActiveWindow.View. Draft = True
ViewDraft, ViewDraft()	x = ActiveWindow.View. Draft
ViewDrawingToolbar	CommandBars("Drawing"). Visible = True ActiveWindow.View. SplitSpecial = wdPaneEndnotes
ViewEndnoteArea, ViewEndnoteArea()	x = ActiveWindow.View. SplitSpecial
ViewEndnoteContNotice	ActiveWindow.View. SplitSpecial = wdPaneEndnoteContinuationNotice
ViewEndnoteContSeparator	ActiveWindow.View. SplitSpecial = wdPaneEndnoteContinuationSeparator
ViewEndnoteSeparator	ActiveWindow.View. SplitSpecial = wdPaneEndnoteSeparator
ViewFieldCodes	ActiveWindow.View. ShowFieldCodes = True ActiveWindow.View. SplitSpecial = wdPaneCurrentPageFooter
	' or
	With ActiveWindow.View
	. Type = wdPrintView
ViewFooter, ViewFooter()	. SeekView = wdSeekCurrentPageFooter
	End With
	' use the StoryType property to return the active story/pane

```

aPane = Selection.StoryType

ActiveWindow.View.SplitSpecial =
ViewFootnoteArea, wdPaneFootnotes
ViewFootnoteArea()

x = ActiveWindow.View.SplitSpecial

ActiveWindow.View.SplitSpecial =
ViewFootnoteContNotice wdPaneFootnoteContinuationNotice

ActiveWindow.View.SplitSpecial =
ViewFootnoteContSeparator wdPaneFootnoteContinuationSeparator

If ActiveDocument.Footnotes.Count >= 1 Then

    ActiveWindow.View.SplitSpecial =
    wdPaneFootnotes

ElseIf ActiveDocument.Endnotes.Count >= 1
Then

    ActiveWindow.View.SplitSpecial =
    wdPaneEndnotes

End If

' Use the Information property to determine if the
selection is in a footnote or endnote pane

x =
Selection.Information(wdInFootnoteEndnotePane)

ActiveWindow.View.SplitSpecial =
ViewFootnoteSeparator wdPaneFootnoteSeparator

ActiveWindow.View.SplitSpecial =
wdPaneCurrentPageHeader

' or

With ActiveWindow.View

```

ViewHeader, ViewHeader() .Type = wdPrintView
 .[SeekView](#) = wdSeekCurrentPageHeader

End With

' use the StoryType property to return the active story/pane

aPane = Selection.[StoryType](#)

ViewMasterDocument,
ViewMasterDocument() ActiveWindow.View.[Type](#) = wdMasterView
 aView = ActiveWindow.View.[Type](#)

ViewMenus() Not applicable in Word 2002
 ActiveWindow.View.[Type](#) = wdNormalView

ViewNormal, ViewNormal() x = ActiveWindow.View.[Type](#)

ViewOutline, ViewOutline() Windows(1).View.[Type](#) = wdOutlineView
 x = Windows(1).View.[Type](#)

ViewPage, ViewPage() Windows(*name*).View.[Type](#) = wdPrintView
 x = Windows(*name*).View.[Type](#)

ViewRibbon, ViewRibbon() CommandBars("Formatting").[Visible](#) = True
 x = CommandBars("Formatting").[Visible](#)

ViewRuler, ViewRuler() ActiveWindow.[DisplayRulers](#) = True
 x = ActiveWindow.[DisplayRulers](#)

ViewStatusBar,
ViewStatusBar() Application.[DisplayStatusBar](#) = True
 x = Application.[DisplayStatusBar](#)

```

If ActiveWindow.View.Type = wdOutlineView
Then

    ActiveWindow.View.Type = wdMasterView

ViewToggleMasterDocument ElseIf ActiveWindow.View.Type =
wdMasterView Then

    ActiveWindow.View.Type = wdOutlineView

End If

With CommandBars

    .LargeButtons = True

    .DisplayToolTips = True

    .DisplayKeysInToolTips = True

ViewToolbars LargeButtons, ToolTips, ToolTipsKey, Reset,
Delete, Show End With

CommandBars(name).Reset

CommandBars(name).Delete

CommandBars(name).Visible = True

Windows(name).View.Zoom.PageFit =
wdPageFitBestFit

With ActiveWindow.View.Zoom

    .PageColumns = 2

ViewZoom AutoFit    .PageRows = 1

ViewZoom TwoPages End With

ViewZoom FullPage ActiveWindow.View.Zoom.PageFit =
wdPageFitFullPage

```

ViewZoom NumColumns,
NumRows

With ActiveWindow.View.Zoom

[.PageColumns](#) = num

ViewZoom ZoomPercent

[.PageRows](#) = num

End With

ActiveWindow.View.Zoom.[Percentage](#) = num

ViewZoom100

Windows(1).View.Zoom.[Percentage](#) = 100

ViewZoom200

ActiveWindow.View.Zoom.[Percentage](#) = 200

ViewZoom75

ActiveWindow.View.Zoom.[Percentage](#) = 75

ViewZoomPageWidth

Windows(name).View.Zoom.[PageFit](#) =
wdPageFitBestFit

ViewZoomWholePage

ActiveWindow.View.Zoom.[PageFit](#) =
wdPageFitFullPage

ActiveWindow.[SmallScroll](#) Down:=num

VLine

' or

ActiveWindow.[SmallScroll](#) Up:=num

ActiveWindow.[LargeScroll](#) Down:=num

VPage

' or

ActiveWindow.[LargeScroll](#) Up:=num

ActiveWindow.[VerticalPercentScrolled](#) = num

VScroll, VScroll()

num = ActiveWindow.[VerticalPercentScrolled](#)



Visual Basic Equivalents W Through Y

A B C D E F G H I J K L M N O P R S T U V W Y

W

WaitCursor	System. Cursor = <i>WdCursorType</i>
Weekday(<i>date</i>)	Weekday(<i>date</i>)
While...Wend	While...Wend
<i>num</i> = Window()	<i>num</i> = ActiveWindow. Index
WindowArrangeAll	Windows. Arrange ArrangeStyle:=wdTiled
WindowList <i>num</i>	Window(<i>num</i>). Activate
WindowName\$()	aCap = ActiveWindow. Caption Windows. Add
WindowNewWindow	' or ActiveWindow. NewWindow
Window <i>num</i>	Window(<i>num</i>). Activate Use the Split property to determine if a Window is split.
WindowPane()	Use StoryType property with the Selection object to determine the pane/story of the selection.
WinToDOS\$()	<i>x</i> = WordBasic . [WinToDOS\$] (<i>StringToTranslate</i>) Selection. MoveLeft Unit:=wdWord, Count:=1, Extend:=wdMove
WordLeft <i>count</i>	Selection. MoveStart Unit:=wdWord, Count:=-1

WordLeft *count, select* ' or

Selection.[MoveLeft](#) Unit:=WdWord, Count:=1,
Extend:=wdExtend

Selection.[MoveRight](#) Unit:=wdWord, Count:=1,
Extend:=wdMove

WordRight 1

Selection.[MoveEnd](#) Unit:=wdWord, Count:=1

WordRight 1, 1

' or

Selection.[MoveRight](#) Unit:=WdWord, Count:=1,
Extend:=wdExtend

WordUnderline,
WordUnderline()

Selection.Range.[Underline](#) = wdUnderlineWords
status = Selection.Range.[Underline](#)

Write

Write

Y

Year Year()



Conceptual Differences Between WordBasic and Visual Basic

The primary difference between Visual Basic for Applications and Microsoft WordBasic is that whereas the WordBasic language consists of a flat list of approximately 900 commands, Visual Basic consists of a [hierarchy](#) of objects, each of which exposes a specific set of methods and properties (similar to statements and functions in WordBasic). While most WordBasic commands can be run at any time, Visual Basic only exposes the methods and properties of the available objects at a given time.

Objects are the fundamental building block of Visual Basic; almost everything you do in Visual Basic involves modifying objects. Every element of Word — documents, paragraphs, fields, bookmarks, and so on — can be represented by an object in Visual Basic. Unlike commands in a flat list, there are objects that can only be accessed from other objects. For example, the [Font](#) object can be accessed from various objects including the [Style](#), [Selection](#), and [Find](#) object.

The programming task of applying bold formatting demonstrates the differences between the two programming languages. The following WordBasic instruction applies bold formatting to the selection.

```
Bold 1
```

The following example is the Visual Basic equivalent for applying bold formatting to the selection.

```
Selection.Font.Bold = True
```

Visual Basic doesn't include a **Bold** statement and function. Instead, there's a property named [Bold](#). (A property is usually an attribute of an object, such as its size, its color, or whether or not it's bold.) **Bold** is a property of the **Font** object. Likewise, [Font](#) is a property of the **Selection** object that returns a **Font** object. Following the object hierarchy, you can build the instruction to apply bold

formatting to the selection.

The **Bold** property is a read/write Boolean property. This means that the **Bold** property can be set to **True** or **False** (on or off), or the current value can be returned. The following WordBasic instruction returns a value indicating whether bold formatting is applied to the selection.

```
x = Bold()
```

The following example is the Visual Basic equivalent for returning the bold formatting status from the selection.

```
x = Selection.Font.Bold
```

The Visual Basic thought process

To perform a task in Visual Basic, you need to determine the appropriate object. For example, if you want to apply character formatting found in the **Font** dialog box, use the **Font** object. Then you need to determine how to "drill down" through the Word object hierarchy from the [Application](#) object to the **Font** object, through the objects that contain the **Font** object you want to modify. After you have determined the path to your object (for example, `Selection.Font`), use the Object Browser, Help, or the features such as Auto List Members in the Visual Basic Editor to determine what properties and methods can be applied to the object. For more information about drilling down to objects using properties and methods, see [Understanding objects, properties, and methods](#).

Properties and methods are often available to multiple objects in the Word object hierarchy. For example, the following instruction applies bold formatting to the entire document.

```
ActiveDocument.Content.Bold = True
```

Also, objects themselves often exist in more than one place in the object hierarchy. For an illustration of the Word object model, see [Microsoft Word Objects](#).

If you know the WordBasic command for the task you want to accomplish in Word 2002, see [Visual Basic Equivalents for WordBasic Commands](#).

The Selection and Range objects

Most WordBasic commands modify the selection. For example, the **Bold** command formats the selection with bold formatting. The **InsertField** command inserts a field at the insertion point. Anytime you want to work with the selection in Visual Basic, you use the [Selection](#) property to return the **Selection** object. The selection can be a block of text or just the insertion point.

The following Visual Basic example inserts text and a new paragraph after the selection.

```
Selection.InsertAfter Text:="Hello World"  
Selection.InsertParagraphAfter
```

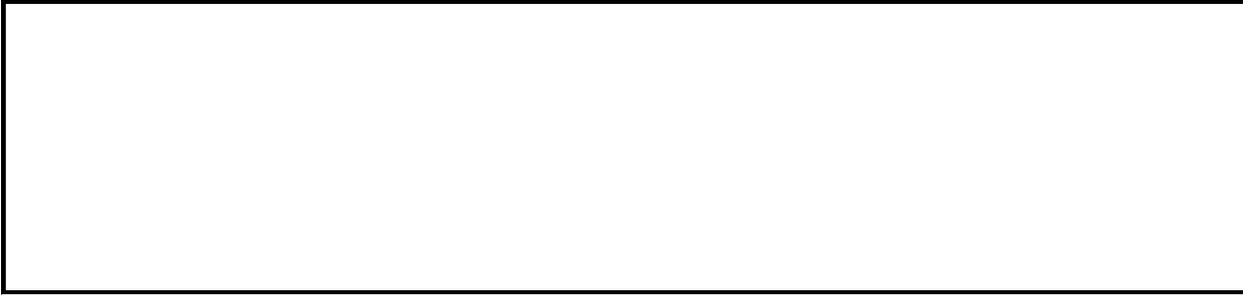
In addition to working with the selection, you can define and work with various ranges of text in a document. A [Range](#) object refers to a contiguous area in a document with a starting character position and ending character position. Similar to the way bookmarks are used in a document, **Range** objects are used in Visual Basic to identify portions of a document. However, unlike a bookmark, a **Range** object is invisible to the user unless the **Range** has been selected using the [Select](#) method. For example, you can use Visual Basic to apply bold formatting anywhere in the document without changing the selection. The following example applies bold formatting to the first 10 characters in the active document.

```
ActiveDocument.Range(Start:=0, End:=10).Bold = True
```

The following example applies bold formatting to the first paragraph.

```
ActiveDocument.Paragraphs(1).Range.Bold = True
```

Both of these example change the formatting in the active document without changing the selection. For more information on the **Range** object see [Working with Range objects](#).



Revising Recorded Visual Basic Macros

The macro recorder is a great tool for discovering the Visual Basic methods and properties you want to use. If you don't know what properties or methods to use, turn on the macro recorder and manually perform the action. The macro recorder translates your actions into Visual Basic code. There are, however, some limitations to recording macros. You cannot record the following:

- Conditional branches
- Variable assignments
- Looping structures
- Custom user forms
- Error handling
- Text selections made with the mouse (you must use keyboard combinations).

To enhance your macros, you may want to revise the code recorded into your module.

Removing the Selection property

Macros created using the macro recorder depend on the selection. At the beginning of most recorded macro instructions, you'll see "Selection." Recorded macros use the [Selection](#) property to return the [Selection](#) object. For example, the following example moves the selection to the Temp bookmark and inserts text after the bookmark.

```
Sub Macro1()  
    Selection.Goto What:=wdGotoBookmark, Name:="Temp"  
    Selection.MoveRight Unit:=wdCharacter, Count:=1  
    Selection.TypeText Text:="New text"  
End Sub
```

This macro accomplishes the task, but there are a couple of drawbacks. First, if the document doesn't have a bookmark named Temp, the macro posts an error. Second, the macro moves the selection, which may not be appropriate. Both of these issues can be resolved by revising the macro so that it doesn't use the **Selection** object. This is the revised macro.

```
Sub MyMacro()  
If ActiveDocument.Bookmarks.Exists("Temp") = True Then  
    endloc = ActiveDocument.Bookmarks("Temp").End  
    ActiveDocument.Range(Start:=endloc, _  
        End:=endloc).InsertAfter "New text"  
End If  
End Sub
```

The [Exists](#) method is used to check for the existence of the bookmark named Temp. If the bookmark is found, the bookmark's ending character position is returned by the [End](#) property. Finally, the [Range](#) method is used to return a [Range](#) object that refers to the bookmark's ending position, so that text can be inserted using the [InsertAfter](#) method. For more information on defining **Range** objects, see [Working with Range objects](#).

Using With...End With

Macro instructions that refer to the same object can be simplified using a **With...End With** structure. For example, the following macro was recorded when a title was added at the top of a document.

```
Sub Macro1()  
    Selection.HomeKey Unit:=wdStory  
    Selection.TypeText Text:="Title"  
    Selection.ParagraphAlignment.Alignment = wdAlignParagraphCenter  
End Sub
```

The **Selection** property is used with each instruction to return a **Selection** object. The macro can be simplified so that the **Selection** property is only used once.

```
Sub MyMacro()  
    With Selection  
        .HomeKey Unit:=wdStory  
        .TypeText Text:="Title"  
        .ParagraphAlignment.Alignment = wdAlignParagraphCenter  
    End With  
End Sub
```

The same task can also be performed without using the **Selection** object. The following macro uses a **Range** object at the beginning of the active document to accomplish the same task.

```
Sub MyMacro()  
    With ActiveDocument.Range(Start:=0, End:=0)  
        .InsertAfter "Title"  
        .ParagraphFormat.Alignment = wdAlignParagraphCenter  
    End With  
End Sub
```

Removing unnecessary properties

If you record a macro that involves selecting an option in a dialog box, the macro recorder records the settings of all the options in the dialog box, even if you only change one or two options. If you don't need to change all the options, you can remove the unnecessary properties from the recorded macro. The following recorded macro includes a number of options from the **Paragraph** dialog box (**Format** menu).

```
Sub Macro1()  
  With Selection.ParagraphFormat  
    .LeftIndent = InchesToPoints(0)  
    .RightIndent = InchesToPoints(0)  
    .SpaceBefore = 6  
    .SpaceAfter = 6  
    .LineSpacingRule = 0  
    .Alignment = wdAlignParagraphLeft  
    .WidowControl = True  
    .KeepWithNext = False  
    .KeepTogether = False  
    .PageBreakBefore = False  
    .NoLineNumber = False  
    .Hyphenation = True  
    .FirstLineIndent = InchesToPoints(0)  
    .OutlineLevel = 10  
  End With  
End Sub
```

However, if you only want to change the spacing before and after the paragraph, you can change the macro to the following.

```
Sub MyMacro()  
  With Selection.ParagraphFormat  
    .SpaceBefore = 6  
    .SpaceAfter = 6  
  End With  
End Sub
```

The simplified macro executes faster because it sets fewer properties. Only the spacing before and after are changed; all the other settings for the selected paragraphs are unchanged.

Removing unnecessary arguments

When the macro recorder records a method, the values of all the arguments are included. The following macro was recorded when the document named Test.doc was opened. The resulting macro includes all the arguments for the [Open](#) method.

```
Sub Macro1()  
    Documents.Open FileName:="C:\My Documents\Test.doc", _  
        ConfirmConversions:= False, ReadOnly:=False, _  
        AddToRecentFiles:=False, PasswordDocument:="", _  
        PasswordTemplate:="", Revert:=False, _  
        WritePasswordDocument:="", _  
        WritePasswordTemplate:="", Format:=wdOpenFormatAuto  
End Sub
```

The arguments that are not needed can be removed from the recorded macro. For example, you could remove all of arguments set to an empty string (for example, `WritePasswordDocument:=""`), as shown.

```
Sub MyMacro()  
    Documents.Open FileName:="C:\My Documents\Test.doc", _  
        ConfirmConversions:= False, _  
        ReadOnly:=False, AddToRecentFiles:=False, _  
        Revert:=False, Format:=wdOpenFormatAuto  
End Sub
```



Adding Controls to a Document

You can add [controls](#) to a document's drawing layer or text layer. To add a control to the drawing layer, click the control on the **Control Toolbox**. Drag an adjustment handle of the control until the control's outline is the size and shape you want. To add a control to the text layer, hold down the SHIFT key while you click a control on the **Control Toolbox**. The control is automatically added to the document at the insertion point.

Note Dragging a control (or a number of "grouped" controls) from the form back to the **Control Toolbox** creates a template of that control, which can be reused. This is a useful feature for implementing a standard interface for your applications.



Auto Macros

By giving a macro a special name, you can run it automatically when you perform an operation such as starting Microsoft Word or opening a document. Word recognizes the following names as automatic macros, or "auto" macros.

Macro name	When it runs
AutoExec	When you start Word or load a global template
AutoNew	Each time you create a new document
AutoOpen	Each time you open an existing document
AutoClose	Each time you close a document
AutoExit	When you quit Word or unload a global template

Auto macros in code modules are recognized if either of the following conditions are true.

- The module is named after the auto macro (for example, AutoExec) and it contains a procedure named "Main."
- A procedure in any module is named after the auto macro.

Just like other macros, auto macros can be stored in the Normal template, another template, or a document. In order for an auto macro to run, it must be either in the Normal template, in the active document, or in the template on which the active document is based. The only exception is the AutoExec macro, which will not run automatically unless it is stored in one of the following: the Normal template, a template that is loaded globally through the **Templates and Add-Ins** dialog box, or a global template stored in the folder specified as the Startup folder.

In the case of a naming conflict (multiple auto macros with the same name), Word runs the auto macro stored in the closest context. For example, if you create an AutoClose macro in a document and the attached template, only the auto macro stored in the document will execute. If you create an AutoNew macro in the normal template, the macro will run if a macro named AutoNew

doesn't exist in the document or the attached template.

Note You can hold down the SHIFT key to prevent auto macros from running. For example, if you create a new document based on a template that contains an AutoNew macro, you can prevent the AutoNew macro from running by holding down SHIFT when you click **OK** in the **New** dialog box (**File** menu) and continuing to hold down SHIFT until the new document is displayed. In a macro that might trigger an auto macro, you can use the following instruction to prevent auto macros from running.

```
WordBasic.DisableAutoMacros
```



Getting Help on Macintosh Keywords

You have requested Help for a Visual Basic keyword used only on the Macintosh. For information about this keyword, consult the language reference Help included with Microsoft Office Macintosh Edition.



Creating a Custom Dialog Box

Use the following procedure to create a custom dialog box:

1. [Create a UserForm](#)

On the **Insert** menu in the Visual Basic Editor, click **UserForm**.

2. [Add controls to the UserForm](#)

Find the control you want to add in the **Toolbox** and drag the control onto the form.

3. [Set control properties](#)

Right-click a control in design mode and click **Properties** to display the Properties window.

4. [Initialize the controls](#)

You can initialize controls in a procedure before you show a form, or you can add code to the Initialize event of the form.

5. [Write event procedures](#)

All controls have a predefined set of events. For example, a command button has a Click event that occurs when the user clicks the command button. You can write event procedures that run when the events occur.

6. [Show the dialog box](#)

Use the **Show** method to display a UserForm.

7. [Use control values while code is running](#)

Some properties can be set at run time. Values the user sets for controls in

the dialog box are lost when the dialog box is closed.



Adding Controls to a UserForm

To add [controls](#) to a user form, find the control you want to add in the **Toolbox**, drag the control onto the form, and then drag an adjustment handle on the control until the control's outline is the size and shape you want.

Note Dragging a control (or a number of "grouped" controls) from the form back to the **Toolbox** creates a template of that control, which can be reused. This is a useful feature for implementing a standard interface for your applications.

When you've added controls to the form, use the commands on the **Format** menu in the Visual Basic Editor to adjust the control alignment and spacing.



UnProtect Method

-
Removes protection from the specified document. If the document isn't protected, this method generates an error.

expression.**UnProtect**(*Password*)

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

Password Optional **Variant**. The password string used to protect the document. Passwords are case-sensitive. If the document is protected with a password and the correct password isn't supplied, a dialog box prompts the user for the password.

Example

This example unprotects the active document, using "Blue" as the password. If the document has a password, a dialog box prompts the user for the password.

```
If ActiveDocument.ProtectionType <> wdNoProtection Then
    ActiveDocument.Unprotect Password:="Blue"
End If
```

This example unprotects the active document. Text is inserted, and the document is protected for revisions.

```
Set aDoc = ActiveDocument
If aDoc.ProtectionType <> wdNoProtection Then
    aDoc.Unprotect
    Selection.InsertBefore "department six"
    aDoc.Protect Type:=wdAllowOnlyRevisions, Password:="Blue"
End If
```



ActiveX Controls

For more information about a specific control, select an object from the following list. For information about events, select a control and click Events at the top of the topic.

[CheckBox](#)

[MultiPage](#)

[ComboBox](#)

[OptionButton](#)

[CommandButton](#)

[ScrollBar](#)

[Frame](#)

[SpinButton](#)

[Image](#)

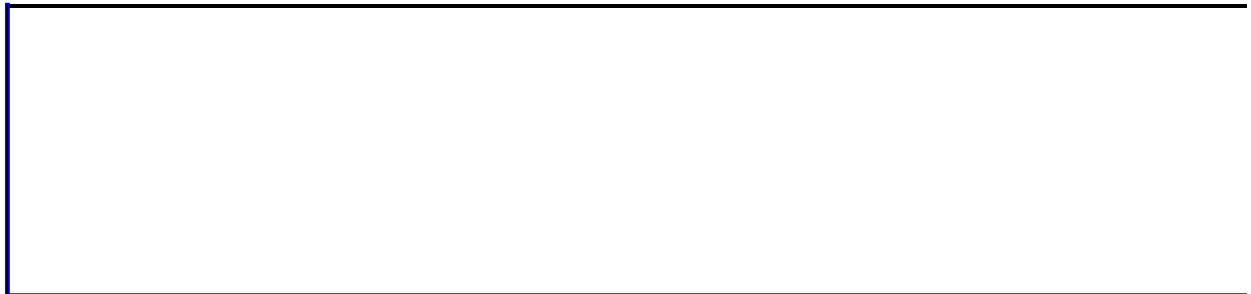
[TabStrip](#)

[Label](#)

[TextBox](#)

[ListBox](#)

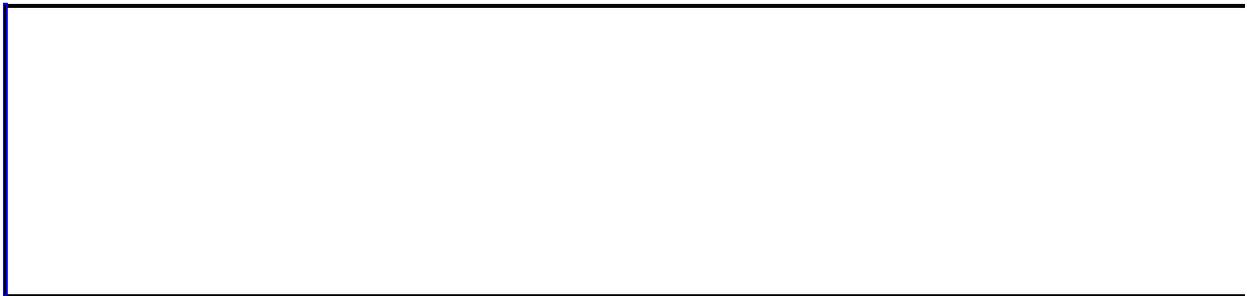
[ToggleButton](#)



Creating a UserForm

To create a custom dialog box, you must create a UserForm. To create a UserForm, click **UserForm** on the **Insert** menu in the Visual Basic Editor.

Use the Properties window to change the name, behavior, and appearance of the form. For example, to change the caption on a form, set the [Caption](#) property.



↳ [Show All](#)

Setting Control Properties

You can set some [control](#) properties at design time (before any macro is running). In [design mode](#), right-click a control and click **Properties** to display the Properties window. Property names are shown in the left column in the window, property values in the right column. You set a property value by entering the new value to the right of the property name.



Initializing Control Properties

You can initialize [controls](#) at run time by using Visual Basic code in a macro. For example, you could fill a list box, set text values, or set option buttons.

The following example uses the Visual Basic **AddItem** method to add data to a list box named `lstRegions`. Then it sets the value of a text box and displays the form.

```
Private Sub GetUserName()  
    With UserForm1  
        .lstRegions.AddItem "North"  
        .lstRegions.AddItem "South"  
        .lstRegions.AddItem "East"  
        .lstRegions.AddItem "West"  
        .txtSalesPersonID.Text = "00000"  
        .Show  
        ' ...  
    End With  
End Sub
```

You can also use code in the Visual Basic Initialize event of a form to set initial values for controls on the form. An advantage to setting initial control values in the Initialize event is that the initialization code stays with the form. You can copy the form to another project, and when you run the **Show** method to display the dialog box, the controls will be initialized.

```
Private Sub UserForm_Initialize()  
    With UserForm1  
        With .lstRegions  
            .AddItem "North"  
            .AddItem "South"  
            .AddItem "East"  
            .AddItem "West"  
        End With  
        .txtSalesPersonID.Text = "00000"  
    End With  
End Sub
```



Control and Dialog Box Events

After you have added [controls](#) to your dialog box or document, you add event procedures to determine how the controls respond to user actions.

UserForms and controls have a predefined set of events. For example, a command button has a Click event that occurs when the user clicks the command button, and UserForms have an **Initialize** event that runs when the form is loaded.

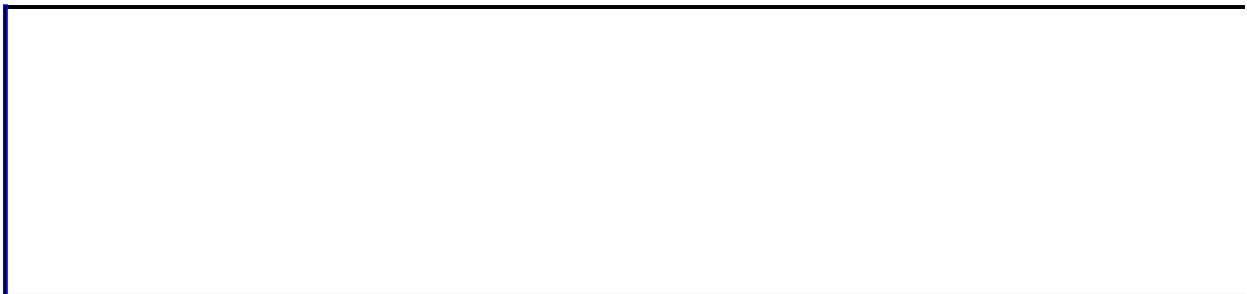
To write a control or form event procedure, open a module by double-clicking the form or control, and select the event from the **Procedure** drop-down list box.

Event procedures include the name of the control. For example, the name of the Click event procedure for a command button named Command1 is Command1_Click.

If you add code to an event procedure and then change the name of the control, your code remains in procedures with the previous name.

For example, assume you add code to the Click event for Command1 and then rename the control to Command2. When you double-click Command2, you will not see any code in the Click event procedure. You will need to move code from Command1_Click to Command2_Click.

To simplify development, it is a good practice to name your controls before writing code.



Displaying a Custom Dialog Box

To test your dialog box in the Visual Basic Editor, click **Run Sub/UserForm** on the **Run** menu.

To display a dialog box from Visual Basic, use the [Show](#) method. The following example displays the dialog box named UserForm1.

```
Private Sub GetUserName()  
    UserForm1.Show  
End Sub
```

Note Use the **Unload** method in an event procedure, such as the Click event procedure for a command button, to close a dialog box.



Using Control Values While Code is Running

Some [controls](#) properties can be set and returned while Visual Basic code is running. The following example sets the **Text** property of a text box to "Hello."

```
TextBox1.Text = "Hello"
```

The data entered in a form by a user is lost when the form is closed. If you return the values of controls on a form after the form has been unloaded, you get the initial values for the controls rather than the values the user entered.

If you want to save the data entered in a form, you can save the information to module-level variables while the form is still running. The following example displays a form and saves the form data in public variables prior to unloading the form.

```
'Code in module to declare public variables
Public strRegion As String
Public intSalesPersonID As Integer
Public blnCancelled As Boolean

'Code in form
Private Sub cmdCancel_Click()
    Module1.blnCancelled = True
    Unload Me
End Sub

Private Sub cmdOK_Click()
    'Save data
    intSalesPersonID = txtSalesPersonID.Text
    strRegion = lstRegions.List(lstRegions.ListIndex)
    Module1.blnCancelled = False
    Unload Me
End Sub

Private Sub UserForm_Initialize()
    Module1.blnCancelled = True
```

End Sub

'Code in module to display form

Sub LaunchSalesPersonForm()

 frmSalesPeople.Show

 If blnCancelled = True Then

 MsgBox "Operation Cancelled!", vbExclamation

 Else

 MsgBox "The Salesperson's ID is: " & _

 intSalesPersonID & _

 "The Region is: " & strRegion

 End If

End Sub

