Action Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActionA"}

Returns or sets the type of action that will occur when the specified shape is clicked or the mouse pointer is positioned over the shape during a slide show. Can be one of the following **PpActionType** constants: **ppActionEndShow**, **ppActionFirstSlide**, **ppActionHyperlink**, **ppActionLastSlide**, **ppActionLastSlideViewed**, **ppActionMixed**, **ppActionNamedSlideShow**, **ppActionNextSlide**, **ppActionNone**, **ppActionOLEVerb**, **ppActionPreviousSlide**, **ppActionRunMacro**, or **ppActionRunProgram**. Read/write **Long**.

You can use the **Action** property in conjunction with other properties of the **ActionSetting** object, as shown in the following table.

If you set the Action property to this value	Use this property	To do this
ppActionHyperlink	**Hyperlink**	Set properties for the hyperlink that will be followed in response to a mouse action on the shape during a slide show.
ppActionRunProgram	**Run**	Return or set the name of the program to run in response to a mouse action on the shape during a slide show.
ppActionRunMacro	**Run**	Return or set the name of the macro to run in response to a mouse action on the shape during a slide show.
ppActionOLEVerb	**ActionVerb**	Set the OLE verb that will be invoked in response to a mouse action on the shape during a slide show.
ppActionNamedSlideShow	**SlideShowName**	Set the name of the custom slide show that will run in response to a mouse action on the shape during a slide show.

Action Property Example

This example sets shape three on slide one in the active presentation to be played when the mouse passes over it during a slide show.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseOver)
    .ActionVerb = "Play"
    .Action = ppActionOLEVerb
End With
```

AnimateAction Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAnimateActionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAnimateActionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAnimateActionA"}

True if the color of the specified shape is momentarily inverted when the specified mouse action occurs. Read/write **Long**.

AnimateAction Property Example

This example sets shape three on slide one in the active presentation to play the sound of applause and to momentarily invert its color when it's clicked during a slide show.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseClick)
    .SoundEffect.Name = "applause.wav"
    .AnimateAction = True
End With
```

Run Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRunC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRunX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRunA"}

Returns or sets the name of the presentation or macro to be run when the specified shape is clicked or the mouse pointer passes over the shape during a slide show. The **Action** property must be set to **ppActionRunMacro** or **ppActionRunProgram** for this property to affect the slide show action. Read/write **String**.

Remarks

If the value of the **Action** property is **ppActionRunMacro**, the specified string value should be the name of a global macro that's currently loaded. If the value of the **Action** property is **ppActionRunProgram**, the specified string value should be the full path and file name of a program.

You can set the **Run** property to a macro that takes no arguments or a macro that takes a single **Shape** or **Object** argument. The shape that was clicked during the slide show will be passed as this argument.

Run Property Example

This example specifies that the CalculateTotal macro be run whenever the mouse pointer passes over the shape during a slide show.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseOver)
    .Action = ppActionRunMacro
    .Run = "CalculateTotal"
    .AnimateAction = True
End With
```

Add Method (AddIns Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddAddInsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddAddInsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddAddInsObjA"}

Adds a new add-in file to the list of add-ins in the **Add-Ins** dialog box (**Tools** menu). Returns an **AddIn** object that represents the newly added add-in.

Syntax

expression.**Add(*Filename*)**

expression Required. An expression that returns an **AddIns** object.

Filename Required **String**. The full name of the file (including the path and file name extension) that contains the add-in you want to add to the list of add-ins.

Remarks

This method doesn't load the new add-in. You must set the **Loaded** property to load the add-in.

Add Method (AddIns Collection Object) Example

This example adds MyTools.ppa to the list in the **Add-Ins** dialog box (**Tools** menu).

```
Set myAddIn = Application.AddIns.Add("c:\my documents\mytools.ppa")
MsgBox myAddIn.Name & " has been added to the list"
```

AutoLoad Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoLoadC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoLoadX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoLoadA"}

True if the specified add-in is automatically loaded each time PowerPoint is started. Read/write **Long**.

Remarks

Setting this property to **True** automatically sets the **Registered** property to **True**.

AutoLoad Property Example

This example displays the name of each add-in that's automatically loaded each time PowerPoint is started.

```
For Each myAddIn In AddIns
    If myAddIn.AutoLoad Then
        MsgBox myAddIn.Name
        afound = True
    End If
Next myAddIn
If afound <> True Then MsgBox "No add-ins were loaded automatically."
```

This example specifies that the add-in named "MyTools" be loaded automatically each time PowerPoint is started.

```
Application.AddIns("mytools").AutoLoad = True
```

Loaded Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLoadedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLoadedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLoadedA"}

True if the specified add-in is loaded. In the **Add-Ins** dialog box (**Tools** menu), the check boxes next to loaded add-ins are selected. Read/write **Long**.

Loaded Property Example

This example adds MyTools.ppa to the list in the **Add-Ins** dialog box (**Tools** menu) and then loads it.

```
Addins.Add("c:\my documents\mytools.ppa").Loaded = True
```

This example unloads the add-in named "MyTools."

```
Application.Addins("mytools").Loaded = False
```

Registered Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproRegisteredC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproRegisteredX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproRegisteredA"}

True if the specified add-in is registered in the system registry. Read/write **Long**.

Remarks

The key for an add-in that's registered by a setup program will usually be located in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\8.0\PowerPoint\Addins*add-in name*. The key for an add-in that you register by setting the **Registered** property to **True** will be located in HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\8.0\PowerPoint\Addins*add-in name*.

Registered Property Example

This example registers the add-in named "MyTools" in HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\8.0\PowerPoint\Addins\MyTools.

```
Application.Addins("MyTools").Registered = True
```

Remove Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRemoveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRemoveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRemoveA"}

Removes an add-in from the collection of add-ins.

Syntax

expression.**Remove(*Index*)**

expression Required. An expression that returns an **AddIns** object.

Index Required **Variant**. The name of the add-in to be removed from the collection.

Remove Method Example

This example removes the add-in named "MyTools" from the list of available add-ins.

```
AddIns.Remove "mytools"
```

AdvanceMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAdvanceModeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAdvanceModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAdvanceModeA"}

AnimationSettings object: Returns or sets a value that indicates whether the specified shape animation advances only when clicked or automatically after a specified amount of time. Can be one of the following **PpAdvanceMode** constants: **ppAdvanceModeMixed**, **ppAdvanceOnClick**, or **ppAdvanceOnTime**. If your shape doesn't become animated, make sure that the **TextLevelEffect** property is set to a value other than **ppAnimateLevelNone** and that the **Animate** property is set to **True**. Read/write **Long**.

SlideShowSettings object: Returns or sets a value that indicates how the slide show advances. Can be one of the following **PpSlideShowAdvanceMode** constants: **ppSlideShowManualAdvance**, **ppSlideShowRehearseNewTimings**, or **ppSlideShowUseSlideTimings**. Read/write **Long**.

SlideShowView object: Returns a value that indicates how the slide show in the specified view advances. Can be one of the following **PpSlideShowAdvanceMode** constants: **ppSlideShowManualAdvance**, **ppSlideShowRehearseNewTimings**, or **ppSlideShowUseSlideTimings**. Read-only **Long**.

AdvanceMode Property Example

This example sets shape two on slide one in the active presentation to become animated automatically after five seconds.

```
With ActivePresentation.Slides(1).Shapes(2).AnimationSettings
    .AdvanceMode = ppAdvanceOnTime
    .AdvanceTime = 5
    .TextLevelEffect = ppAnimateByAllLevels
    .Animate = True
End With
```

AdvanceTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAdvanceTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAdvanceTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAdvanceTimeA"}

AnimationSettings object: Returns or sets the amount of time after which the specified shape will become animated. Read/write **Single**.

SlideShowTransition object: Returns or sets the amount of time after which the specified slide transition will occur. Read/write **Single**.

Remarks

The specified slide animation won't start automatically after the amount of time you've specified unless the **AdvanceMode** property of the animation is set to **ppAdvanceOnTime**. The specified slide transition won't advance automatically unless the **AdvanceMode** property of the slide show settings is set to **ppSlideShowUseSlideTimings**.

AdvanceTime Property Example

This example sets shape two on slide one in the active presentation to become animated automatically after five seconds.

```
With ActivePresentation.Slides(1).Shapes(2).AnimationSettings
    .AdvanceMode = ppAdvanceOnTime
    .AdvanceTime = 5
    .TextLevelEffect = ppAnimateByAllLevels
    .Animate = True
End With
```

AfterEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAfterEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAfterEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAfterEffectA"}

Returns or sets a value that indicates whether the specified shape appears dimmed, hidden, or unchanged after it's been built. Can be one of the following **PpAfterEffect** constants: **ppAfterEffectDim**, **ppAfterEffectHide**, **ppAfterEffectHideOnClick**, **ppAfterEffectMixed**, or **ppAfterEffectNothing** (the default value). Read/write **Long**.

Remarks

You won't see the after effect you set for a shape unless the shape gets animated and at least one other shape on the slide gets animated after it. For a shape to be animated, the **TextLevelEffect** property of the **AnimationSettings** object for the shape must be set to something other than **ppAnimateLevelNone** and the **Animate** property must be set to **True**. To change the build order of the shapes on a slide, use the **AnimationOrder** property.

AfterEffect Property Example

This example specifies that the title on slide two in the active presentation appear dimmed after the title is built. If the title is the last or only shape to be built on slide two, the text won't be dimmed.

```
With ActivePresentation.Slides(1).Shapes.Title.AnimationSettings
    .TextLevelEffect = ppAnimateByAllLevels
    .AfterEffect = ppAfterEffectDim
    .Animate = True
End With
```

Animate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproAnimateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproAnimateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproAnimateA"}

True if the specified shape is animated during a slide show. Read/write **Long**.

Remarks

This property must be set to **True** for any of the other properties of the **AnimationSettings** object to take effect.

Animate Property Example

This example specifies that the title on slide two in the active presentation appear dimmed after the title is built. If the title is the last or only shape to be built on slide two, the text won't be dimmed.

```
With ActivePresentation.Slides(2).Shapes.Title.AnimationSettings
    .TextLevelEffect = ppAnimateByAllLevels
    .AfterEffect = ppAfterEffectDim
    .Animate = True
End With
```

AnimateBackground Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAnimateBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAnimateBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAnimateBackgroundA"}

If the specified object is an AutoShape, **True** if the shape is animated separately from the text it contains. If the specified shape is a graph object, **True** if the background (the axes and gridlines) of the specified graph object is animated. Applies only to AutoShapes with text that can be built in more than one step or to graph objects. Read/write **Long**.

Remarks

Use the **TextLevelEffect** and **TextUnitEffect** properties to control the animation of text attached to the specified shape.

If this property is set to **True** and the **TextLevelEffect** property is set to **ppAnimateByAllLevels**, the shape and its text will be animated simultaneously. If this property is set to **True** and the **TextLevelEffect** property is set to anything other than **ppAnimateByAllLevels**, the shape will be animated immediately before the text is animated.

You won't see effects of setting this property unless the specified shape is animated. For a shape to be animated, the **TextLevelEffect** property of the **AnimationSettings** object for the shape must be set to something other than **ppAnimateLevelNone** and the **Animate** property must be set to **True**.

AnimateBackground Property Example

This example creates a rectangle that contains text. The example then specifies that the shape should fly in from the lower right, that the text should be built from first-level paragraphs, and that the shape should be animated separately from the text it contains.

```
With ActivePresentation.Slides(1).Shapes.AddShape(msoShapeRectangle, 50,
50, 200, 300)
    .TextFrame.TextRange = "Reason 1" & Chr(13) & "Reason 2" & Chr(13) &
"Reason 3"
    With .AnimationSettings
        .EntryEffect = ppEffectFlyFromBottomRight
        .TextLevelEffect = ppAnimateByFirstLevel
        .TextUnitEffect = ppAnimateByParagraph
        .AnimateBackground = True
        .Animate = True
    End With
End With
```

AnimateTextInReverse Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAnimateTextInReverseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAnimateTextInReverseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAnimateTextInReverseA"}

True if the specified shape is built in reverse order. Applies only to shapes (such as shapes containing lists) that can be built in more than one step. Read/write **Long**.

Remarks

You won't see effects of setting this property unless the specified shape gets animated. For a shape to be animated, the **TextLevelEffect** property of the **AnimationSettings** object for the shape must be set to something other than **ppAnimateLevelNone** and the **Animate** property must be set to **True**.

AnimateTextInReverse Property Example

This example adds a slide after slide one in the active presentation, sets the title text, adds a three-item list to the text placeholder, and sets the list to be built in reverse order.

```
With ActivePresentation.Slides.Add(2, ppLayoutText).Shapes
    .Item(1).TextFrame.TextRange.Text = "Top Three Reasons"
    With .Item(2)
        .TextFrame.TextRange = "Reason 1" & Chr(13) & "Reason 2" & Chr(13)
& "Reason 3"
        With .AnimationSettings
            .Animate = True
            .TextLevelEffect = ppAnimateByFirstLevel
            .AnimateTextInReverse = True
        End With
    End With
End With
```

AnimationOrder Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAnimationOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAnimationOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAnimationOrderA"}

Returns or sets an integer that represents the position of the specified shape within the collection of shapes to be animated. Read/write **Long**.

Remarks

You won't see effects of setting this property unless the specified shape gets animated. For a shape to be animated, the **TextLevelEffect** property of the **AnimationSettings** object for the shape must be set to something other than **ppAnimateLevelNone** and the **Animate** property must be set to **True**.

AnimationOrder Property Example

This example specifies that shape two on slide two the active presentation be animated second.

```
ActivePresentation.Slides(2).Shapes(2).AnimationSettings.AnimationOrder = 2
```

ChartUnitEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproChartUnitEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproChartUnitEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproChartUnitEffectA"}

Returns or sets a value that indicates whether the graph range is animated by series, category, or element. Can be one of the following **ChartUnitEffect** values: **ppAnimateByCategory**, **ppAnimateByCategoryElements**, **ppAnimateBySeries**, **ppAnimateBySeriesElements**, or **ppAnimateChartMixed**. Read/write **Long**.

Remarks

If your graph doesn't become animated, make sure that the **Animate** property is set to **True**

ChartUnitEffect Property Example

This example sets shape two on slide three in the active presentation to be animated by series. Shape two must be a graph for this to work.

```
With ActivePresentation.Slides(3).Shapes(2)
    With .AnimationSettings
        .ChartUnitEffect = ppAnimateBySeries
        .EntryEffect = ppEffectFlyFromLeft
        .Animate = True
    End With
End With
```

DimColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDimColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDimColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDimColorA"}

Returns or sets a **ColorFormat** object that represents the color of the specified shape after it's been built. Read-only.

Remarks

If you don't get the effect you expect, check your other build settings. You won't see the effect of the **DimColor** property unless the **TextLevelEffect** property of the **AnimationSettings** object is set to something other than **ppAnimateLevelNone**, the **AfterEffect** property is set to **ppAfterEffectDim**, and the **Animate** property is set to **True**. In addition, if the specified shape is the only item or the last item to be built on the slide, the shape won't be dimmed. To change the build order of the shapes on a slide, use the **AnimationOrder** property.

DimColor Property Example

This example adds a slide that contains both a title and a three-item list to the active presentation, sets the title and list to be dimmed after being built, and sets the color that each of them will be dimmed to.

```
With ActivePresentation.Slides.Add(2, ppLayoutText).Shapes
    With .Item(1)
        .TextFrame.TextRange.Text = "Sample title"
        With .AnimationSettings
            .TextLevelEffect = ppAnimateByAllLevels
            .AfterEffect = ppAfterEffectDim
            .DimColor.SchemeColor = ppShadow
            .Animate = True
        End With
    End With
    With .Item(2)
        .TextFrame.TextRange.Text = "Item one" & Chr(13) & "Item two"
        With .AnimationSettings
            .TextLevelEffect = ppAnimateByFirstLevel
            .AfterEffect = ppAfterEffectDim
            .DimColor.RGB = RGB(100, 150, 130)
            .Animate = True
        End With
    End With
End With
```

EntryEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEntryEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEntryEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEntryEffectA"}

For the **AnimationSettings** object, this property returns or sets the special effect applied to the animation for the specified shape. For the **SlideShowTransition** object, this property returns or sets the special effect applied to the specified slide transition. Read/write **Long**.

Can be one of the following **PpEntryEffect** constants:

ppEffectAppear
ppEffectBlindsHorizontal
ppEffectBlindsVertical
ppEffectBoxIn
ppEffectBoxOut
ppEffectCheckerboardAcross
ppEffectCheckerboardDown
ppEffectCoverDown
ppEffectCoverLeft
ppEffectCoverLeftDown
ppEffectCoverLeftUp
ppEffectCoverRight
ppEffectCoverRightDown
ppEffectCoverRightUp
ppEffectCoverUp
ppEffectCrawlFromDown
ppEffectCrawlFromLeft
ppEffectCrawlFromRight
ppEffectCrawlFromUp
ppEffectCut
ppEffectCutThroughBlack
ppEffectDissolve
ppEffectFade
ppEffectFlashOnceFast
ppEffectFlashOnceMedium
ppEffectFlashOnceSlow
ppEffectFlyFromBottom
ppEffectFlyFromBottomLeft
ppEffectFlyFromBottomRight
ppEffectFlyFromLeft
ppEffectFlyFromRight
ppEffectFlyFromTop
ppEffectFlyFromTopLeft
ppEffectFlyFromTopRight
ppEffectMixed
ppEffectNone
ppEffectPeekFromDown
ppEffectPeekFromLeft
ppEffectPeekFromRight
ppEffectPeekFromUp
ppEffectRandom
ppEffectRandomBarsHorizontal
ppEffectRandomBarsVertical
ppEffectSplitHorizontalIn
ppEffectSplitHorizontalOut
ppEffectSplitVerticalIn
ppEffectSplitVerticalOut
ppEffectStripsDownLeft
ppEffectStripsDownRight
ppEffectStripsLeftDown
ppEffectStripsLeftUp
ppEffectStripsRightDown
ppEffectStripsRightUp
ppEffectStripsUpLeft
ppEffectStripsUpRight
ppEffectUncoverDown
ppEffectUncoverLeft
ppEffectUncoverLeftDown
ppEffectUncoverLeftUp
ppEffectUncoverRight
ppEffectUncoverRightDown
ppEffectUncoverRightUp
ppEffectUncoverUp
ppEffectWipeDown
ppEffectWipeLeft
ppEffectWipeRight
ppEffectWipeUp

Remarks

If the **TextLevelEffect** property for the specified shape is set to **ppAnimateLevelNone** (the default value) or the **Animate** property is set to **False**, you won't see the special effect you've applied with the

EntryEffect property.

EntryEffect Property Example

This example adds a title slide to the active presentation and sets the title to fly in from the right whenever it's animated during a slide show.

```
With ActivePresentation.Slides.Add(1, ppLayoutTitleOnly).Shapes(1)
        .TextFrame.TextRange.Text = "Sample title"
    With .AnimationSettings
        .TextLevelEffect = ppAnimateByAllLevels
        .EntryEffect = ppEffectFlyFromRight
        .Animate = True
    End With
End With
```

PlaySettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPlaySettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPlaySettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPlaySettingsA"}

Returns a **PlaySettings** object that contains information about how the specified media clip plays during a slide show. Read-only.

PlaySettings Property Example

This example inserts a movie named "Clock.avi" onto slide one in the active presentation, sets it to play automatically after the slide transition, and specifies that the movie object be hidden during a slide show except when it's playing.

```
With ActivePresentation.Slides(1).Shapes.AddOLEObject(Left:=10, Top:=10, _
        Width:=250, Height:=250, FileName:="c:\winnt\clock.avi")
    With .AnimationSettings.PlaySettings
        .PlayOnEntry = True
        .HideWhileNotPlaying = True
    End With
End With
```

SoundEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSoundEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSoundEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSoundEffectA"}

ActionSetting object: Returns a **SoundEffect** object that represents the sound to be played when the specified shape is clicked or the mouse pointer passes over the shape. If you don't hear the sound that you assigned to the shape when you run the slide show, make sure that the **TextLevelEffect** property is set to a value other than **ppAnimateLevelNone** and that the **Animate** property is set to **True**. Read-only.

AnimationSettings object: Returns a **SoundEffect** object that represents the sound to be played during the animation of the specified shape. Read-only.

SlideShowTransition object: Returns a **SoundEffect** object that represents the sound to be played during the transition to the specified slide. Read-only.

SoundEffect Property Example

This example sets the file Bass.wav to be played whenever shape one on slide one in the active presentation is animated.

```
With ActivePresentation.Slides(1).Shapes(1).AnimationSettings
    .Animate = True
    .TextLevelEffect = ppAnimateByAllLevels
    .SoundEffect.ImportFromFile "c:\sndsys\bass.wav"
End With
```

TextLevelEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextLevelEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextLevelEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextLevelEffectA"}

Returns or sets a value that indicates whether the text in the specified shape is animated by first-level paragraphs, second-level paragraphs, or some other level (up to fifth-level paragraphs). Can be one of the following **PpTextLevelEffect** constants: **ppAnimateByAllLevels**, **ppAnimateByFifthLevel**, **ppAnimateByFirstLevel**, **ppAnimateByFourthLevel**, **ppAnimateBySecondLevel**, **ppAnimateByThirdLevel**, **ppAnimateLevelMixed**, or **ppAnimateLevelNone**. Read/write **Long**.

Remarks

For the **TextLevelEffect** property setting to take effect, the **Animate** property must be set to **True**.

TextLevelEffect Property Example

This example adds a title slide and title text to the active presentation and sets the title to be built letter by letter.

```
With ActivePresentation.Slides.Add(1, ppLayoutTitleOnly).Shapes(1)
    .TextFrame.TextRange.Text = "Sample title"
    With .AnimationSettings
        .Animate = True
        .TextLevelEffect = ppAnimateByFirstLevel
        .TextUnitEffect = ppAnimateByCharacter
        .EntryEffect = ppEffectFlyFromLeft
    End With
End With
```

TextUnitEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextUnitEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextUnitEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextUnitEffectA"}

Returns or sets a value that indicates whether the text in the specified shape is animated paragraph by paragraph, word by word, or letter by letter. Can be one of the following **TextUnitEffect** values: **ppAnimateByCharacter**, **ppAnimateByParagraph**, **ppAnimateByWord**, or **ppAnimateUnitMixed**. Read/write **Long**.

Remarks

For the **TextUnitEffect** property setting to take effect, the **TextLevelEffect** property for the specified shape must have a value other than **ppAnimateLevelNone** or **ppAnimateByAllLevels**, and the **Animate** property must be set to **True**.

TextUnitEffect Property Example

This example adds a title slide and title text to the active presentation and sets the title to be built letter by letter.

```
With ActivePresentation.Slides.Add(1, ppLayoutTitleOnly).Shapes(1)
    .TextFrame.TextRange.Text = "Sample title"
    With .AnimationSettings
        .Animate = True
        .TextLevelEffect = ppAnimateByFirstLevel
        .TextUnitEffect = ppAnimateByCharacter
        .EntryEffect = ppEffectFlyFromLeft
    End With
End With
```

ActivePresentation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActivePresentationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActivePresentationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActivePresentationA"}

Returns a **Presentation** object that represents the presentation open in the active window. Read-only.

Note that if an embedded presentation is in-place active, the **ActivePresentation** property returns the embedded presentation.

ActivePresentation Property Example

This example adds a new color scheme to the active presentation, sets the title color, and applies the new color scheme to all the slides in the presentation.

```
With Application.ActivePresentation.ColorSchemes.Add
    .Item(ppTitleColor).RGB = RGB(255, 0, 0)
    .ApplyToAllSlides
End With
```

ActivePrinter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActivePrinterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActivePrinterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActivePrinterA"}

Returns the name of the active printer. Read-only **String**.

ActivePrinter Property Example

This example displays the name of the active printer.

```
MsgBox "The name of the active printer is " & Application.ActivePrinter
```

ActiveWindow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActiveWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActiveWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActiveWindowA"}

Returns a **DocumentWindow** object that represents the active document window. Read-only.

ActiveWindow Property Example

This example minimizes the active window.

```
Application.ActiveWindow.WindowState = ppWindowMinimized
```

AddIns Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAddInsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAddInsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAddInsA"}

Returns an **AddIns** collection that represents all the add-ins listed in the **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.

AddIns Property Example

This example adds the add-in named "Myaddin.ppa" to the list in the **Add-Ins** dialog box and installs the add-in automatically.

```
Set myAddIn = Application.AddIns.Add(FileName:="c:\myaddin.ppa")
MsgBox myAddIn.Name & " has been added to the list"
```

Assistant Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAssistantC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAssistantX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAssistantA"}

Returns an **Assistant** object that represents the Office Assistant. Read-only.

Assistant Property Example

This example displays the Office Assistant.

```
Application.Assistant.Visible = True
```

This example moves the Office Assistant to the upper-left region of the screen.

```
Application.Assistant.Move xLeft:=100, yTop:=100
```

Build Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBuildC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBuildX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBuildA"}

Returns the PowerPoint build number. Read-only **String**.

Build Property Example

This example displays the PowerPoint build number.

```
MsgBox Prompt:=Application.Build, Title:="PowerPoint Build"
```

Caption Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCaptionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCaptionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCaptionA"}

Application object: Returns the text that appears in the title bar of the application window. Read-write **String**.

DocumentWindow object: Returns the text that appears in the title bar of the document window. Read-only **String**.

Caption Property Example

This example displays the caption for each open document window.

```
With Application.Windows
    For w = 1 To .Count
        MsgBox "Window " & w & " contains " & .Item(1).Caption
    Next
End With
```

CommandBars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCommandBarsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCommandBarsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCommandBarsA"}

Returns a **CommandBars** collection that represents all the command bars in PowerPoint. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

CommandBars Property Example

This example enlarges all command bar buttons and enables ToolTips.

```
With Application.CommandBars
    .LargeButtons = True
    .DisplayTooltips = True
End With
```

This example displays the **Formatting** command bar at the top of the application window.

```
With Application.CommandBars("Formatting")
    .Visible = True
    .Position = msoBarTop
End With
```

FileSearch Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFileSearchC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFileSearchX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFileSearchA"}

Returns a **FileSearch** object that can be used to search for files using either an absolute or relative path. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Note This property is available only in Microsoft Windows.

FileSearch Property Example

This example displays the names of all files in the My Documents folder that begin with "New."

```
With Application.FileSearch
    .FileName = "New*.*"
    .LookIn = "C:\My Documents"
    .Execute
    For I = 1 To .FoundFiles.Count
        MsgBox .FoundFiles(I)
    Next
End With
```

Help Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthHelpC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthHelpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthHelpA"}

Displays a Help topic.

Syntax

expression.**Help(*HelpFile, ContextID*)**

expression Required. An expression that returns an **Application** object.

HelpFile Optional **String**. The name of the online Help file you want to display. If this argument isn't specified, PowerPoint Help is used.

ContextID Optional **Long**. The context ID number for the Help topic. If this argument isn't specified, the **Help Topics** dialog box is displayed.

Help Method Example

This example displays topic number 65527 in the Help file Otisapp.hlp.

```
Application.Help "OTISAPP.HLP", 65527
```

OperatingSystem Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOperatingSystemC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOperatingSystemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOperatingSystemA"}

Returns the name of the operating system. Read-only **String**.

OperatingSystem Property Example

This example tests the **OperatingSystem** property to see whether PowerPoint is running with a 32-bit version of Microsoft Windows.

```
os = Application.OperatingSystem
If InStr(os, "Windows (32-bit)") <> 0 Then
   MsgBox "Running a 32-bit version of Microsoft Windows"
End If
```

Path Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPathC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPathX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPathA"}

Returns the path to the specified **AddIn**, **Application**, or **Presentation** object. Read-only **String**.

Note The path doesn't include the final separator character (in Windows, this is a backslash (\); on the Macintosh, it's a colon (:)) or the name of the specified object. Use the **Name** property of the **Presentation** object to return the file name without the path, and use the **FullName** property to return the file name and the path together.

Path Property Example

This example saves the active presentation in the same folder as PowerPoint.

```
With Application
    fName = .Path & "\test presentation"
    ActivePresentation.SaveAs fName
End With
```

Presentations Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresentationsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresentationsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresentationsA"}

Returns a **Presentations** collection that represents all open presentations. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Presentations Property Example

This example opens the presentation named "Long Version.ppt."

```
Application.Presentations.Open _
    FileName:="c:\My Documents\Long version.ppt"
```

This example saves presentation one as "Year-End Report.ppt."

```
Application.Presentations(1).SaveAs "Year-End Report"
```

This example closes the Year-end report presentation.

```
Application.Presentations("Year-End Report.ppt").Close
```

Quit Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthQuitC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthQuitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthQuitA"}

Quits PowerPoint. This is equivalent to clicking **Exit** on the **File** menu.

Syntax

expression.**Quit**

expression Required. An expression that returns an **Application** object.

Remarks

To avoid being prompted to save changes, use either the **Save** or **SaveAs** method to save all open presentations before calling the **Quit** method.

Quit Method Example

This example saves all open presentations and then quits PowerPoint.

```
With Application
    For Each w In .Presentations
        w.Save
    Next w
    .Quit
End With
```

SlideShowWindows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideShowWindowsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideShowWindowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideShowWindowsA"}

Returns a **SlideShowWindows** collection that represents all open slide show windows. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

SlideShowWindows Property Example

This example runs a slide show in a window and sets the height and width of the slide show window.

```
With Application
    .Presentations(1).SlideShowSettings.Run
    With .SlideShowWindows(1)
        .Height = 250
        .Width = 250
    End With
End With
```

VBE Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVBAC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVBAX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVBAA"}

Returns a **VBE** object that represents the VB Editor. Read-only.

VBE Property Example

This example sets the name of the active project in the VB Editor.

```
Application.VBE.ActiveVBProject.Name = "TestProject"
```

Version Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVersionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVersionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVersionA"}

Returns the PowerPoint version number. Read-only **String**.

Version Property Example

This example displays a message box that contains the PowerPoint version number and build number, and the name of the operating system.

```
With Application
MsgBox "Welcome to PowerPoint version " & .Version & _
    ", build " & .Build & ", running on " & .OperatingSystem & "!"
End With
```

Windows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproWindowsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproWindowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproWindowsA"}

Application object: Returns a **DocumentWindows** collection that represents all open document windows. Read-only.

Presentation object: Returns a **DocumentWindows** collection that represents all document windows associated with the specified presentation. This property doesn't return any slide show windows associated with the presentation. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Windows Property Example

This example closes all windows except the active window.

```
With Application.Windows
    For i = 2 To .Count
        .Item(2).Close
    Next
End With
```

Character Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCharacterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCharacterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCharacterA"}

Returns or sets the Unicode character value that is used for bullets in the specified text. Read/write **Long**.

Character Property Example

This example sets the bullet character for shape two on slide one in the active presentation.

```
Set frame2 = ActivePresentation.Slides(1).Shapes(2).TextFrame
With frame2.TextRange.ParagraphFormat.Bullet
    .Character = 8226
    .Visible = True
End With
```

Font Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontA"}

Returns a **Font** object that represents character formatting. Read-only.

Font Property Example

This example sets the formatting for the text in shape one on slide one in the active presentation.

```
With ActivePresentation.Slides(1).Shapes(1)
    With .TextFrame.TextRange.Font
        .Size = 48
        .Name = "Palatino"
        .Bold = True
        .Color.RGB = RGB(255, 127, 255)
    End With
End With
```

This example sets the color and font name for bullets in shape two on slide one.

```
With ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat.Bullet
        .Visible = True
        With .Font
            .Name = "Palatino"
            .Color.RGB = RGB(0, 0, 255)
        End With
    End With
End With
```

RelativeSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproRelativeSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproRelativeSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproRelativeSizeA"}

Returns or sets the bullet size relative to the size of the first text character in the paragraph. Can be a floating-point value from 0.25 through 4, indicating that the bullet size can be from 25 percent through 400 percent of the text-character size. Read/write **Single**.

RelativeSize Property Example

This example sets the formatting for the bullet in shape two on slide one in the active presentation. The size of the bullet is 125 percent of the size of the first text character in the paragraph.

```
With ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat.Bullet
        .Visible = True
        .RelativeSize = 1.25
        .Character = 169
        With .Font
            .Name = "Symbol"
            .Color.RGB = RGB(255, 0, 0)
        End With
    End With
End With
```

UseTextColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproUseTextColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproUseTextColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproUseTextColorA"}

True if the specified bullets are set to the color of the first text character in the paragraph. **False** if the specified bullets are set to any other color. Read/write **Long**.

Remarks

You cannot explicitly set this property to **False**. Setting the bullet format color (using the **Color** property of the **Font** object) sets this property to **False**. When **UseTextColor** is **False**, you can set it to **True** to reset the bullet format to the default color.

UseTextColor Property Example

This example resets bullets in shape two on slide one in the active presentation to their default character, font, and color.

```
With ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat.Bullet
        .RelativeSize = 1
        .UseTextColor = True
        .UseTextFont = True
        .Character = 8226
    End With
End With
```

UseTextFont Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproUseTextFontC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproUseTextFontX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproUseTextFontA"}

True if the specified bullets are set to the font of the first text character in the paragraph. **False** if the specified bullets are set to a custom font. Read/write **Long**.

Remarks

You cannot explicitly set this property to **False**. Setting the bullet format font (using the **Name** property of the **Font** object) sets this property to **False**. When **UseTextFont** is **False**, you can set it to **True** to reset the bullet format to the default font.

UseTextFont Property Example

This example resets bullets in shape two on slide one in the active presentation to their default character, font, and color.

```
With ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat.Bullet
        .RelativeSize = 1
        .UseTextColor = True
        .UseTextFont = True
        .Character = 8226
    End With
End With
```

Accent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro AccentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro AccentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro AccentA"}

True if a vertical accent bar separates the callout text from the callout line. Read/write **Long**.

Accent Property 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 = ActivePresentation.Slides(1)
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
```

Angle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAngleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAngleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAngleA"}

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. Can be one of the following **MsoCalloutAngleType** constants: **msoCalloutAngle30**, **msoCalloutAngle45**, **msoCalloutAngle60**, **msoCalloutAngle90**, **msoCalloutAngleAutomatic**, or **msoCalloutAngleMixed**. Read/write **Long**.

Remarks

If you set the value of this property to anything other than **msoCalloutAngleAutomatic**, the callout line maintains a fixed angle as you drag the callout.

Angle Property Example

This example sets to 90 degrees the callout angle for a callout named "co1" on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes("co1").Callout.Angle = msoCalloutAngle90
```

AutoAttach Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproAutoAttachC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproAutoAttachX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproAutoAttachA"}

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 **Long**.

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.

AutoAttach Property Example

This example adds two callouts to `myDocument`. 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    With .AddCallout(msoCalloutTwo, 420, 170, 200, 50)
        .TextFrame.TextRange.Text = "auto-attached"
        .Callout.AutoAttach = True
    End With
    With .AddCallout(msoCalloutTwo, 420, 350, 200, 50)
        .TextFrame.TextRange.Text = "not auto-attached"
        .Callout.AutoAttach = False
    End With
End With
```

AutoLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoLengthA"}

True if the first segment of the callout line (the segment attached to the text callout box) is scaled automatically whenever the callout is moved. **False** if the first segment of the callout retains the fixed length specified 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**). Read-only **Long**.

Remarks

This property is read-only. Use the **AutomaticLength** method to set this property to **True**, and use the **CustomLength** method to set this property to **False**.

AutoLength Property Example

This example toggles between an automatically scaling first segment and one with a fixed length for the callout line for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Callout
    If .AutoLength Then
        .CustomLength 50
    Else
        .AutomaticLength
    End If
End With
```

AutomaticLength Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAutomaticLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAutomaticLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAutomaticLengthA"}

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**).

Syntax

expression.**AutomaticLength**

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

Remarks

Applying this method sets the **AutoLength** property to **True**.

AutomaticLength Method Example

This example toggles between an automatically scaling first segment and one with a fixed length for the callout line for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Callout
    If .AutoLength Then
        .CustomLength 50
    Else
        .AutomaticLength
    End If
End With
```

Border Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BorderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BorderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BorderA"}

True if the text in the specified callout is surrounded by a border. Read/write **Long**.

Border Property 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 = ActivePresentation.Slides(1)
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
```

CustomDrop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCustomDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCustomDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCustomDropA"}

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.

Syntax

expression.**CustomDrop(*Drop*)**

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

Drop Required **Single**. The drop distance, in points.

CustomDrop Method Example

This example sets the custom drop distance to 14 points, and specifies that the drop distance always be measured from the top. For the example to work, shape three must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Callout
    .CustomDrop 14
    .AutoAttach = False
End With
```

CustomLength Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCustomLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCustomLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCustomLengthA"}

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**).

Syntax

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.

CustomLength Method Example

This example toggles between an automatically scaling first segment and one with a fixed length for the callout line for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Callout
    If .AutoLength Then
        .CustomLength 50
    Else
        .AutomaticLength
    End If
End With
```

Drop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDropA"}

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**.

Drop Property Example

This example replaces the custom drop for shape one on `myDocument` 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, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Callout
    If .DropType = msoCalloutDropCustom Then
        If .Drop < .Parent.Height / 2 Then
            .PresetDrop msoCalloutDropTop
        Else
            .PresetDrop msoCalloutDropBottom
        End If
    End If
End With
```

DropType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproDropTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproDropTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproDropTypeA"}

Returns a value that indicates where the callout line attaches to the callout text box. Can be one of the following **MsoCalloutDropType** constants: **msoCalloutDropBottom**, **msoCalloutDropCenter**, **msoCalloutDropCustom**, **msoCalloutDropMixed**, or **msoCalloutDropTop**. Read-only **Long**.

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.

DropType Property Example

This example checks to determine whether shape three on `myDocument` 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

Gap Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGapC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGapX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGapA"}

Returns or sets the horizontal distance (in points) between the end of the callout line and the text bounding box. Read/write **Single**.

Gap Property Example

This example sets the distance between the callout line and the text bounding box to 3 points for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).Callout.Gap = 3
```

Length Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLengthA"}

CalloutFormat object: 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**.

TextRange object: Returns the length of the specified text range, in characters. Read-only **Long**.

Remarks

This property is read-only. Use the **CustomLength** method to set the value of this property for the **CalloutFormat** object.

Length Property Example

If the first line segment in the callout named "co1" has a fixed length, this example specifies that the length of the first line segment in the callout named "co2" will also be fixed at that length. For the example to work, both callouts must have multiple-segment lines.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    With .Item("co1").Callout
        If Not .AutoLength Then len1 = .Length
    End With
    If len1 Then .Item("co2").Callout.CustomLength len1
End With
```

This example sets the title font size to 48 points if the title of slide two contains more than five characters, or it sets the font size to 72 points if the title has five or fewer characters.

```
Set myDocument = ActivePresentation.Slides(2)
With myDocument.Shapes(1).TextFrame.TextRange
    If .Length > 5 Then
        .Font.Size = 48
    Else
        .Font.Size = 72
    End If
End With
```

PresetDrop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPresetDropC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPresetDropX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPresetDropA"}

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.

Syntax

expression.**PresetDrop(*DropType*)**

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

DropType Required **Long**. The starting position of the callout line relative to the text bounding box. Can be one of the following **MsoCalloutDropType** constants: **msoCalloutDropBottom**, **msoCalloutDropCenter**, or **msoCalloutDropTop**. Specifying **msoCalloutDropCustom** for this argument will cause your code to fail.

PresetDrop Method Example

This example specifies that the callout line attach to the top of the text bounding box for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).Callout.PresetDrop msoCalloutDropTop
```

This example toggles between two preset drops for shape one on `myDocument`. For the example to work, shape one must be a callout.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Callout
    If .DropType = msoCalloutDropTop Then
        .PresetDrop msoCalloutDropBottom
    ElseIf .DropType = msoCalloutDropBottom Then
        .PresetDrop msoCalloutDropTop
    End If
End With
```

RGB Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRGBC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRGBX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRGBA"}

ColorFormat object: Returns or sets the red-green-blue (RGB) value of the specified color. Read/write **Long**.

RGBColor object: Returns or sets the red-green-blue (RGB) value of the specified color-scheme color or extra color. Read/write **Long**.

RGB Property Example

This example sets the background color for color scheme three in the active presentation and then applies the color scheme to all slides in the presentation that are based on the slide master.

```
With ActivePresentation
    Set cs1 = .ColorSchemes(3)
    cs1.Colors(ppBackground).RGB = RGB(128, 128, 0)
    .SlideMaster.ColorScheme = cs1
End With
```

This example displays the value of the red, green, and blue components of the fill forecolor for shape one on slide one in the active document.

```
Set myDocument = ActivePresentation.Slides(1)
c = myDocument.Shapes(1).Fill.ForeColor.RGB
redComponent = c Mod 256
greenComponent = c \ 256 Mod 256
blueComponent = c \ 65536 Mod 256
MsgBox "RGB components: " & redComponent & ", " & greenComponent & ", " &
blueComponent
```

RGB value

Value returned by the **RGB** function; specifies a color as a combination of red, green, and blue values. The **RGB** function has the following syntax:

RGB(*red, green, blue*)

The arguments specify the red, green, and blue components of the color as integers from 0 to 255.

SchemeColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSchemeColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSchemeColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSchemeColorA"}

Returns or sets the color in the applied color scheme that's associated with the specified object. Can be one of the following **PpColorSchemeIndex** constants: **ppAccent1**, **ppAccent2**, **ppAccent3**, **ppBackground**, **ppFill**, **ppForeground**, **ppNotSchemeColor**, **ppSchemeColorMixed**, **ppShadow**, or **ppTitle**. Read/write **Long**.

SchemeColor Property Example

This example switches the background color on slide one in the active presentation between an explicit red-green-blue value and the color-scheme background color.

```
With ActivePresentation.Slides(1)
    .FollowMasterBackground = False
    With .Background.Fill.ForeColor
        If .Type = msoColorTypeScheme Then
            .RGB = RGB(0, 128, 128)
        Else
            .SchemeColor = ppBackground
        End If
    End With
End With
```

Add Method (ColorSchemes Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddColorSchemesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddColorSchemesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddColorSchemesObjA"}

Adds a color scheme to the collection of available schemes. Returns a **ColorScheme** object that represents the added color scheme.

Syntax

expression.**Add(*ColorScheme*)**

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

ColorScheme Optional **ColorScheme** object. The color scheme to add. Can be a **ColorScheme** object from any slide or master, or an item in the **ColorSchemes** collection from any open presentation. If this argument is omitted, the first **ColorScheme** object (the first standard color scheme) in the specified presentation's **ColorSchemes** collection is used.

Remarks

The new color scheme is based on the colors used on the specified slide or master or on the colors in the specified color scheme from an open presentation.

The **ColorSchemes** collection can contain up to 16 color schemes. If you need to add another color scheme and the **ColorSchemes** collection is already full, use the **Delete** method to remove an existing color scheme.

Note that although PowerPoint automatically checks whether a color scheme is a duplicate before adding it through the user interface, it doesn't check before adding a color scheme through a Visual Basic procedure. Your procedure must do its own checking to avoid adding redundant color schemes.

Add Method (ColorSchemes Collection Object) Example

This example adds a new color scheme to the collection of standard color schemes for the active presentation. The new color scheme is based on the colors used in slide two in the active presentation.

```
With ActivePresentation
    Set newClrScheme = .Slides(2).ColorScheme
    .ColorSchemes.Add newClrScheme
End With
```

Add Method (ExtraColors Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddExtraColorsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddExtraColorsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddExtraColorsObjA"}

Adds a color to the extra colors available to a presentation if the color hasn't already been added.

Syntax

expression.**Add(*Type*)**

expression Required. An expression that returns an **ExtraColors** object.

Type Required **Long**. The red-green-blue (RGB) value of the color to be added.

Remarks

The **ExtraColors** object can contain only eight colors, including colors added by the user.

extra colors

Nonscheme colors that you use for special purposes. Extra colors appear in color palettes displayed by dialog box controls and toolbar buttons. Changing the color scheme doesn't affect either the available extra colors or objects formatted with extra colors.

Add Method (ExtraColors Object) Example

This example adds an extra color to the active presentation (if the color hasn't already been added).

```
ActivePresentation.ExtraColors.Add RGB(69, 32, 155)
```

Application Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproApplicationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproApplicationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproApplicationA"}

Returns an **Application** object that represents the creator of the specified object. Read-only.

Application Property Example

In this example, a **Presentation** object is passed to the procedure. The procedure adds a slide to the presentation and then saves the presentation in the folder where PowerPoint is running.

```
Sub AddAndSave(pptPres As Presentation)
    pptPres.Slides.Add 1, 1
    pptPres.SaveAs pptPres.Application.Path & "\Added Slide"
End Sub
```

This example displays the name of the application that created each linked OLE object on slide one in the active presentation.

```
For Each s In ActivePresentation.Slides(1).Shapes
    If s.Type = msoLinkedOLEObject Then
        MsgBox s.OLEFormat.Object.Application.Name
    End If
Next
```

Count Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCountA"}

Returns the number of objects in the specified collection. Read-only **Long**.

Count Property Example

This example closes all windows except the active window.

```
With Application.Windows
    For i = 2 To .Count
        .Item(2).Close
    Next
End With
```

Creator Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCreatorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCreatorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCreatorA"}

Returns a 32-bit integer that represents the four-character creator code for the application in which the specified object was created. For example, if the object was created in PowerPoint, this property returns the hexadecimal number 50505433, which represents the string "PPT3". Read-only **Long**.

Remarks

The **Creator** property is designed to be used in Microsoft Office applications for the Macintosh.

Creator Property Example

This example displays a message about the creator of `myObject`.

```
Set myObject = Application.ActivePresentation.Slides(1).Shapes(1)
If myObject.Creator = &h50505433 Then
   MsgBox "This is a PowerPoint object"
Else
   MsgBox "This is not a PowerPoint object"
End If
```

Delete Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDeleteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDeleteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDeleteA"}

Syntax 1: Deletes the specified object.

Syntax 2: Deletes the specified node.

Syntax 3: Deletes the specified tag.

Syntax 1

expression.**Delete**

Syntax 2

expression.**Delete(*Index*)**

Syntax 3

expression.**Delete(*Name*)**

expression Syntax 1: An expression that returns the object to be deleted.

Syntax 2: An expression that returns a **ShapeNodes** object.

Syntax 3: An expression that returns a **Tags** object.

Index Required **Integer**. Specifies the node to be deleted. The segment following that node will also be deleted. If the node is a control point of a curve, the curve and all of its nodes will be deleted.

Name Required **String**. Specifies the name of the tag to be deleted.

Delete Method Example

This example deletes all free-form shapes from slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes
    For s = .Count To 1 Step -1
        With .Item(s)
            If .Type = msoFreeform Then .Delete
        End With
    Next
End With
```

Parent Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproParentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproParentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproParentA"}

Returns the parent object for the specified object. Read-only.

Parent Property Example

This example adds an oval containing text to slide one in the active presentation and rotates the oval and the text 45 degrees. The parent object for the text frame is the **Shape** object that contains the text.

```
Set myShapes = ActivePresentation.Slides(1).Shapes
With myShapes.AddShape(msoShapeOval, 50, 50, 300, 150).TextFrame
    .TextRange.Text = "Test text"
    .Parent.Rotation = 45
End With
```

Tags Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTagsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTagsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTagsA"}

Returns a **Tags** object that represents the tags for the specified object. Read-only.

Tags Property Example

This example adds a tag named "Region" and a tag named "Priority" to slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Tags
    .Add "Region", "East"        'Adds "Region" tag with value "East"
    .Add "Priority", "Low"       'Adds "Priority" tag with value "Low"
End With
```

This example searches through the tags for each slide in the active presentation. If there's a tag named "Priority," a message box displays the tag value. If the object doesn't have a tag named "Priority," the example adds this tag with the value "Unknown."

```
For Each s In Application.ActivePresentation.Slides
    With s.Tags
        found = False
        For i = 1 To .Count
            If .Name(i) = "Priority" Then
                found = True
                slNum = .Parent.SlideIndex
                MsgBox "Slide " & slNum & " priority: " & .Value(i)
            End If
        Next
        If Not found Then
            slNum = .Parent.SlideIndex
            .Add "Priority", "Unknown"
            MsgBox "Slide " & slNum & " priority tag added: Unknown"
        End If
    End With
Next
```

Copy Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCopyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCopyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCopyA"}

Copies the specified object to the Clipboard.

Syntax

expression.**Copy**

expression Required. An expression that returns a **Selection**, **Shape**, **ShapeRange**, **Slide**, **SlideRange**, or **TextRange** object.

Remarks

Use the **Paste** method to paste the contents of the Clipboard.

Copy Method Example

This example copies the selection in window one to the Clipboard and then pastes it into the view in window two. If the Clipboard contents cannot be pasted into the view in window two – for example, if you try to paste a shape into slide sorter view – this example fails.

```
Windows(1).Selection.Copy
Windows(2).View.Paste
```

This example copies shapes one and two on slide one in the active presentation to the Clipboard and then pastes the copies onto slide two.

```
With ActivePresentation
    .Slides(1).Shapes.Range(Array(1, 2)).Copy
    .Slides(2).Shapes.Paste
End With
```

This example copies slide one in the active presentation to the Clipboard.

```
ActivePresentation.Slides(1).Copy
```

This example copies the text in shape one on slide one in the active presentation to the Clipboard.

```
ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange.Copy
```

Cut Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCutA"}

Deletes the specified object and places it on the Clipboard.

Syntax

expression.**Cut**

expression Required. An expression that returns a **Selection**, **Shape**, **ShapeRange**, **Slide**, **SlideRange**, or **TextRange** object.

Cut Method Example

This example deletes the selection in window one and places a copy of it on the Clipboard.

```
Windows(1).Selection.Cut
```

This example deletes shapes one and two from slide one in the active presentation, places copies of them on the Clipboard, and then pastes the copies onto slide two.

```
With ActivePresentation
    .Slides(1).Shapes.Range(Array(1, 2)).Cut
    .Slides(2).Shapes.Paste
End With
```

This example deletes slide one from the active presentation and places a copy of it on the Clipboard.

```
ActivePresentation.Slides(1).Cut
```

This example deletes the text in shape one on slide one in the active presentation and places a copy of it on the Clipboard.

```
ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange.Cut
```

Duplicate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDuplicateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDuplicateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDuplicateA"}

Shape object: Creates a duplicate of the specified **Shape** object, adds the new range of shapes to the **Shapes** collection immediately after the shape specified originally, and then returns the new **Shape** object.

ShapeRange object: Creates a duplicate of the specified **ShapeRange** object, adds the new range of shapes to the **Shapes** collection immediately after the range of shapes specified originally, and then returns the new **ShapeRange** object.

Slide object: Creates a duplicate of the specified **Slide** object, adds the new slide to the **Slides** collection immediately after the slide specified originally, and then returns a **SlideRange** object that represents the duplicate slide.

SlideRange object: Creates a duplicate of the specified **SlideRange** object, adds the new range of slides to the **Slides** collection immediately after the slide range specified originally, and then returns a **SlideRange** object that represents the duplicate slides.

Syntax

expression.**Duplicate**

expression Required. An expression that returns a **Shape**, **ShapeRange**, **Slide**, or **SlideRange** object.

Duplicate Method Example

This example creates a duplicate of slide one in the active presentation and then sets the background shading and the title text of the new slide. The new slide will be slide two in the presentation.

```
Set newSlide = ActivePresentation.Slides(1).Duplicate
With newSlide
    .Background.Fill.PresetGradient msoGradientVertical, 1, msoGradientGold
    .Shapes.Title.TextFrame.TextRange.Text = "Second Quarter Earnings"
End With
```

This example adds a new, blank slide at the end of the active presentation, adds a diamond shape to the new slide, duplicates the diamond, and then sets properties for the duplicate. The first diamond will have the default fill color for the active color scheme; the second diamond will be offset from the first one and will have the default shadow color.

```
Set mySlides = ActivePresentation.Slides
Set newSlide = mySlides.Add(mySlides.Count + 1, ppLayoutBlank)
Set firstObj = newSlide.Shapes.AddShape(msoShapeDiamond, 10, 10, 250, 350)
With firstObj.Duplicate
    .Left = 150
    .Fill.ForeColor.SchemeColor = ppShadow
End With
```

Height Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHeightA"}

Returns or sets the height of the specified object, in points. Read-only **Single** for the **Master** object, read/write **Single** for all other objects.

Remarks

The **Height** property of a **Shape** object returns or sets the height of the forward-facing surface of the specified shape. This measurement doesn't include shadows or 3-D effects.

Height Property Example

This example sets the height of document window two to half the height of the application window.

```
Windows(2).Height = Application.Height / 2
```

Left Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLeftA"}

Returns or sets the left position of the object (in points), as shown in the following table. Read/write **Single**.

Object	Meaning
Application, **SlideShowWindow**	The distance from the left edge of the application window or slideshow window to the left edge of the desktop. Setting this property to a very large positive or negative value may position the window completely off the desktop.
DocumentWindow	The distance from the left edge of the document window to the left edge of the application window's client area.
Shape	The distance from the left edge of the shape's bounding box to the left edge of the document.
ShapeRange	The distance from the left edge of the leftmost shape in the shape range to the left edge of the document.

Left Property Example

This example arranges windows one and two horizontally; in other words, each window occupies half the available vertical space and all the available horizontal space in the application window's client area. For this example to work, there must be only two document windows open.

```
Windows.Arrange ppArrangeTiled
ah = Windows(1).Height                      ' available height
aw = Windows(1).Width + Windows(2).Width    ' available width
With Windows(1)
    .Width = aw
    .Height = ah / 2
    .Left = 0
End With
With Windows(2)
    .Width = aw
    .Height = ah / 2
    .Top = ah / 2
    .Left = 0
End With
```

Name Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNameA"}

Returns or sets the name of the specified object. You can use the object's name in conjunction with the **Item** method to return a reference to the object if the **Item** method for the collection that contains the object takes a **Variant** argument. For example, if the value of the **Name** property for a shape is "Rectangle 2," `Shapes("Rectangle 1")` will return a reference to that shape. The following table contains information about the **Name** property for each object it applies to.

Object	Name property details
AddIn	The name of the add-in includes the file name extension (for file types that are registered) but doesn't include its path. Read-only **String**.
Application	Returns the string "Microsoft PowerPoint." Read-only **String**.
Font	Read/write **String**.
Master	Read/write **String**.
NamedSlideShow	You cannot use this property to set the name for a custom slide show. Use the **Add** method to redefine a custom slide show under a new name. Read-only **String**.
Presentation	The name of the presentation includes the file name extension (for file types that are registered) but doesn't include its path. You cannot use this property to set the name. Use the **SaveAs** method to save the presentation under a different name if you need to change the name. Read-only **String**.
Shape or **ShapeRange** (the shape range must contain exactly one shape)	When a shape is created, Powerpoint automatically assigns it a name in the form *ShapeType Number*, where *ShapeType* identifies the type of shape or AutoShape, and *Number* is an integer that's unique within the collection of shapes on the slide. For example, the automatically generated names of the shapes on a slide could be "Placeholder 1," "Oval 2," and "Rectangle 3." To avoid conflict with automatically assigned names, don't use the form *ShapeType Number* for user-defined names, where *ShapeType* is a value that is used for automatically generated names, and *Number* is any positive integer. Read/write **String**.
Slide or **SlideRange** (the slide range must contain exactly one slide)	When a slide is inserted into a presentation, Powerpoint automatically assigns it a name in the form Slide*n*, where *n* is an integer that represents the order in which the slide was created in the presentation. For example, the first slide inserted into a presentation is automatically named "Slide1." If you copy a slide from one presentation to another, the slide loses the name it had in the first presentation and is automatically assigned a new name in the second presentation. Read/write **String**.
SoundEffect	The set of valid names for a presentation appears

in the the **Sound** box in the **Slide Transition** dialog box (**Slide Show** menu). Read/write **String**.

Name Property Example

This example sets the name of object two on slide one in the active presentation to "big triangle."

```
ActivePresentation.Slides(1).Shapes(2).Name = "big triangle"
```

This example sets the fill color for the shape named "big triangle" on slide one in the active presentation.

```
ActivePresentation.Slides(1).Shapes("big triangle").Fill.ForeColor.RGB =
RGB(0, 0, 255)
```

Paste Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPasteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPasteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPasteA"}

Shapes object: Pastes the shapes, slides, or text on the Clipboard into the specified **Shapes** collection, at the top of the z-order. Each pasted object becomes a member of the specified **Shapes** collection. If the Clipboard contains entire slides, the slides will be pasted as shapes that contain the images of the slides. If the Clipboard contains a text range, the text will be pasted into a newly created **TextFrame** shape. Returns a **ShapeRange** object that represents the pasted objects.

Slides object: Pastes the slides on the Clipboard into the **Slides** collection for the presentation. Specify where you want to insert the slides with the ***Index*** argument. Returns a **SlideRange** object that represents the pasted objects. Each pasted slide becomes a member of the specified **Slides** collection.

TextRange object: Pastes the text on the Clipboard into the specified text range, and returns a **TextRange** object that represents the pasted text.

View object: Pastes the contents of the Clipboard into the active view. Attempting to paste an object into a view that won't accept it causes an error. For information about views and the objects you can paste into them, see the "Remarks" section.

Syntax 1

expression.**Paste**

Syntax 2

expression.**Paste(*Index*)**

expression Required. An expression that returns a **Shapes**, **TextRange**, or **View** object (Syntax 1) or a **Slides** object (Syntax 2).

Index Optional **Long**. The index number of the slide that the slides on the Clipboard are to be pasted before. If this argument is omitted, the slides on the Clipboard are pasted after the last slide in the presentation.

Remarks

Use the **ViewType** property to set the view for a window before pasting the Clipboard contents into it. The following table shows what you can paste into each view.

Into this view	You can paste the following from the Clipboard
Slide view or notes page view	Shapes, text, or entire slides. If you paste a slide from the Clipboard, an image of the slide will be inserted onto the slide, master, or notes page as an embedded object. If one shape is selected, the pasted text will be appended to the shape's text; if text is selected, the pasted text will replace the selection; if anything else is selected, the pasted text will be placed in it's own text frame. Pasted shapes will be added to the top of the z-order and won't replace selected shapes.
Outline view	Text or entire slides. You cannot paste shapes into outline view. A pasted slide will be inserted before the slide that contains the insertion point.
Slide sorter view	Entire slides. You cannot paste shapes or text into slide sorter view. A pasted slide will be inserted at the insertion point or after the last slide selected in the presentation.

Paste Method Example

This example copies the selection in window one to the Clipboard and copies it into the view in window two. If the Clipboard contents cannot be pasted into the view in window two — for example, if you try to paste a shape into slide sorter view — this example fails.

```
Windows(1).Selection.Copy
Windows(2).View.Paste
```

This example copies the selection in window one to the Clipboard, makes sure that window one is in slide view, and then copies the Clipboard contents into the view in window two.

```
Windows(1).Selection.Copy
With Windows(2)
    .ViewType = ppViewSlide
    .View.Paste
End With
```

This example copies shape one on slide one in the active presentation to the Clipboard and then pastes it into slide two.

```
With ActivePresentation
    .Slides(1).Shapes(1).Copy
    .Slides(2).Shapes.Paste
End With
```

This example cuts the text in shape one on slide one in the active presentation, places it on the Clipboard, and then pastes it after the first word in shape two on the same slide.

```
With ActivePresentation.Slides(1)
    .Shapes(1).TextFrame.TextRange.Cut
    .Shapes(2).TextFrame.TextRange.Words(1).InsertAfter.Paste
End With
```

This example cuts slides three and five from the Old Sales presentation and then inserts them before slide four in the active presentation.

```
Presentations("Old Sales").Slides.Range(Array(3, 5)).Cut
ActivePresentation.Slides.Paste 4
```

Top Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTopA"}

Returns or sets the top position of the object (in points), as shown in the following table. Read/write **Single**.

Object	Meaning
Application, **SlideShowWindow**	The distance from the top edge of the application window or slide show window to the top edge of the desktop. Setting this property to a very large positive or negative value may position the window completely off the desktop.
DocumentWindow	The distance from the top edge of the document window to the top edge of the application window's client area.
Shape	The distance from the top edge of the shape's bounding box to the top edge of the document.
ShapeRange	The distance from the top edge of the topmost shape in the shape range to the top edge of the document. Read-only.

Top Property Example

This example arranges windows one and two horizontally; in other words, each window occupies half the available vertical space and all the available horizontal space in the application window's client area. For this example to work, there must be only two document windows open.

```
Windows.Arrange ppArrangeTiled
ah = Windows(1).Height                      ' available height
aw = Windows(1).Width + Windows(2).Width    ' available width
With Windows(1)
    .Width = aw
    .Height = ah / 2
    .Left = 0
End With
With Windows(2)
    .Width = aw
    .Height = ah / 2
    .Top = ah / 2
    .Left = 0
End With
```

Type Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTypeA"}

Returns and (for some objects) sets the type of object. The following table lists all the objects the **Type** property applies to and all the possible values for the **Type** property of each of those objects.

Object	Type property details
CalloutFormat	Can be one of the following **MsoCalloutType** constants: , **msoCalloutFour**, **msoCalloutMixed**, **msoCalloutOne**, **msoCalloutThree**, or **msoCalloutTwo**. Read/write **Long**.
ColorFormat	Can be one of the following **MsoColorType** constants: **msoColorTypeMixed**, **msoColorTypeRGB**, or **msoColorTypeScheme**. Read-only **Long**.
ConnectorFormat	Can be one of the following **MsoConnectorType** constants: **msoConnectorCurve**, **msoConnectorElbow**, **msoConnectorStraight**, or **msoConnectorTypeMixed**. Read/write **Long**.
FillFormat	Can be one of the following **MsoFillType** constants: **msoFillBackground**, **msoFillGradient**, **msoFillMixed**, **msoFillPatterned**, **msoFillPicture**, **msoFillSolid**, or **msoFillTextured**. Read-only **Long**.
Hyperlink	Can be one of the following **MsoHyperlinkType** constants: **msoHyperlinkInlineShape** (for use in Microsoft Word only), **msoHyperlinkRange**, or **msoHyperlinkShape**. Read-only **Long**.
PlaceholderFormat	Can be one of the following **PpPlaceholderType** constants: **ppPlaceholderBitmap**, **ppPlaceholderBody**, **ppPlaceholderCenterTitle**, **ppPlaceholderChart**, **ppPlaceholderDate**, **ppPlaceholderFooter**, **ppPlaceholderHeader**, **ppPlaceholderMediaClip**, **ppPlaceholderMixed**, **ppPlaceholderObject**, **ppPlaceholderOrgChart**, **ppPlaceholderSlideNumber**, **ppPlaceholderSubtitle**, **ppPlaceholderTable**, **ppPlaceholderTitle**, **ppPlaceholderVerticalBody**, or **ppPlaceholderVerticalTitle**
Selection	Can be one of the following **PpSelectionType** constants: **ppSelectionNone**, **ppSelectionShapes**, **ppSelectionSlides**, or **ppSelectionText**. Read-only **Long**.
ShadowFormat	Can be one of the following **MsoShadowType** constants: **msoShadow1**, **msoShadow10**, **msoShadow11**, **msoShadow12**, **msoShadow13**, **msoShadow14**, **msoShadow15**, **msoShadow16**, **msoShadow17**, **msoShadow18**, **msoShadow19**, **msoShadow2**, **msoShadow20**, **msoShadow3**, **msoShadow4**, **msoShadow5**, **msoShadow6**, **msoShadow7**, **msoShadow8**, **msoShadow9**, or **msoShadowMixed**. Read/write **Long**.
Shape or **ShapeRange**	Can be one of the following **MsoShapeType** constants: **msoAutoShape**, **msoCallout**, **msoChart**,

	msoComment, **msoEmbeddedOLEObject**, **msoFormControl**, **msoFreeform**, **msoGroup**, **msoLine**, **msoLinkedOLEObject**, **msoLinkedPicture**, **msoMedia**, **msoOLEControlObject**, **msoPicture**, **msoPlaceholder**, **msoShapeTypeMixed**, **msoTextBox**, or **msoTextEffect**. Read-only **Long**.
SoundEffect	Can be one of the following **PpSoundEffectType** constants: **ppSoundEffectsMixed**, **ppSoundFile**, **ppSoundNone**, or **ppSoundStopPrevious**. Read/write **Long**.
TabStop	Can be one of the following **PpTabStopType** constants: **ppTabStopCenter**, **ppTabStopDecimal**, **ppTabStopLeft**, **ppTabStopMixed**, or **ppTabStopRight**. Read/write **Long**.
View	Can be one of the following **PpViewType** constants: **ppViewHandoutMaster**, **ppViewNotesMaster**, **ppViewNotesPage**, **ppViewOutline**, **ppViewSlide**, **ppViewSlideMaster**, **ppViewSlideSorter**, or **ppViewTitleMaster**. Read-only **Long**.

Type Property Example

This example loops through all the shapes on all the slides in the active presentation and sets all linked Microsoft Excel worksheets to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Excel.Sheet.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

Visible Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVisibleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVisibleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVisibleA"}

True if the specified object, or the formatting applied to the specified object, is visible. Read/write **Long**.

Visible Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```

Width Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproWidthA"}

Returns or sets the width of the specified object, in points. Read-only **Single** for the **Master** object, read/write **Single** for all other objects.

Width Property Example

This example arranges windows one and two horizontally; in other words,each window occupies half the available vertical space and all the available horizontal space in the application window's client area. For this example to work, there must be only two document windows open.

```
Windows.Arrange ppArrangeTiled
ah = Windows(1).Height                      ' available height
aw = Windows(1).Width + Windows(2).Width    ' available width
With Windows(1)
    .Width = aw
    .Height = ah / 2
    .Left = 0
End With
With Windows(2)
    .Width = aw
    .Height = ah / 2
    .Top = ah / 2
    .Left = 0
End With
```

BeginConnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthBeginConnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthBeginConnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthBeginConnectA"}

Attaches the beginning of the specified connector to a specified shape. If there's already a connection between the beginning of the connector and another shape, that connection is broken. If the beginning of the connector isn't already positioned at the specified connecting site, this method moves the beginning of the connector to the connecting site and adjusts the size and position of the connector. Use the **EndConnect** method to attach the end of the connector to a shape.

Syntax

expression.**BeginConnect(*ConnectedShape, ConnectionSite*)**

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

ConnectedShape Required **Shape** object. The shape to attach the beginning of the connector to. The specified **Shape** object must be in the same **Shapes** collection as the connector.

ConnectionSite Required **Long**. A connection site on the shape specified by ***ConnectedShape***. Must be an integer between 1 and the integer returned by the **ConnectionSiteCount** property of the specified shape. If you want the connector to automatically find the shortest path between the two shapes it connects, specify any valid integer for this argument and then use the **RerouteConnections** method after the connector is attached to shapes at both ends.

Remarks

When you attach a connector to an object, the size and position of the connector are automatically adjusted, if necessary.

BeginConnect Method Example

This example adds two rectangles to `myDocument` and connects them with a curved connector. Notice that the **RerouteConnections** method makes it irrelevant what values you supply for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 100, 100).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    .Parent.RerouteConnections
End With
```

BeginConnected Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BeginConnectedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BeginConnectedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BeginConnectedA"}

True if the beginning of the specified connector is connected to a shape. Read-only **Long**.

BeginConnected Property Example

If shape three on `myDocument` is a connector whose beginning is connected to a shape, this example stores the connection site number in the variable `oldBeginConnSite`, stores a reference to the connected shape in the object variable `oldBeginConnShape`, and then disconnects the beginning of the connector from the shape.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Connector Then
        With .ConnectorFormat
            If .BeginConnected Then
                oldBeginConnSite = .BeginConnectionSite
                Set oldBeginConnShape = .BeginConnectedShape
                .BeginDisconnect
            End If
        End With
    End If
End With
```

BeginConnectedShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BeginConnectedShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BeginConnectedShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BeginConnectedShapeA"}

Returns a **Shape** object that represents the shape that the beginning of the specified connector is attached to. Read-only.

Note If the beginning of the specified connector isn't attached to a shape, this property generates an error.

BeginConnectedShape Property Example

This example assumes that `myDocument` already contains two shapes attached by a connector named "Conn1To2." The code adds a rectangle and a connector to `myDocument`. The beginning of the new connector will be attached to the same connection site as the beginning of the connector named "Conn1To2," and the end of the new connector will be attached to connection site one on the new rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    Set r3 = .AddShape(msoShapeRectangle, 450, 190, 200, 100)
    .AddConnector(msoConnectorCurve, 0, 0, 10, 10).Name = "Conn1To3"
    With .Item("Conn1To2").ConnectorFormat
        beginConnSite1 = .BeginConnectionSite
        Set beginConnShape1 = .BeginConnectedShape
    End With
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect beginConnShape1, beginConnSite1
        .EndConnect r3, 1
    End With
End With
```

BeginConnectionSite Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BeginConnectionSiteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BeginConnectionSiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BeginConnectionSiteA"}

Returns an integer that specifies the connection site that the beginning of a connector is connected to. Read-only **Long**.

Note If the beginning of the specified connector isn't attached to a shape, this property generates an error.

BeginConnectionSite Property Example

This example assumes that `myDocument` already contains two shapes attached by a connector named "Conn1To2." The code adds a rectangle and a connector to `myDocument`. The beginning of the new connector will be attached to the same connection site as the beginning of the connector named "Conn1To2," and the end of the new connector will be attached to connection site one on the new rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    Set r3 = .AddShape(msoShapeRectangle, 450, 190, 200, 100)
    .AddConnector(msoConnectorCurve, 0, 0, 10, 10).Name = "Conn1To3"
    With .Item("Conn1To2").ConnectorFormat
        beginConnSite1 = .BeginConnectionSite
        Set beginConnShape1 = .BeginConnectedShape
    End With
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect beginConnShape1, beginConnSite1
        .EndConnect r3, 1
    End With
End With
```

BeginDisconnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthBeginDisconnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthBeginDisconnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthBeginDisconnectA"}

Detaches the beginning of the specified connector from the shape it's attached to. This method doesn't alter the size or position of the connector: the beginning of the connector remains positioned at a connection site but is no longer connected. Use the **EndDisconnect** method to detach the end of the connector from a shape.

Syntax

expression.**BeginDisconnect**

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

BeginDisconnect Method Example

This example adds two rectangles to `myDocument`, attaches them with a connector, automatically reroutes the connector along the shortest path, and then detaches the connector from the rectangles.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 0, 0).ConnectorFormat
    .BeginConnect firstRect, 1
    .EndConnect secondRect, 1
    .Parent.RerouteConnections
    .BeginDisconnect
    .EndDisconnect
End With
```

EndConnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthEndConnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthEndConnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthEndConnectA"}

Attaches the end of the specified connector to a specified shape. If there's already a connection between the end of the connector and another shape, that connection is broken. If the end of the connector isn't already positioned at the specified connecting site, this method moves the end of the connector to the connecting site and adjusts the size and position of the connector. Use the **BeginConnect** method to attach the beginning of the connector to a shape.

Syntax

expression.**EndConnect(*ConnectedShape, ConnectionSite*)**

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

ConnectedShape Required **Shape** object. The shape to attach the end of the connector to. The specified **Shape** object must be in the same **Shapes** collection as the connector.

ConnectionSite Required **Long**. A connection site on the shape specified by ***ConnectedShape***. Must be an integer between 1 and the integer returned by the **ConnectionSiteCount** property of the specified shape. If you want the connector to automatically find the shortest path between the two shapes it connects, specify any valid integer for this argument and then use the **RerouteConnections** method after the connector is attached to shapes at both ends.

Remarks

When you attach a connector to an object, the size and position of the connector are automatically adjusted, if necessary.

EndConnect Method Example

This example adds two rectangles to `myDocument` and connects them with a curved connector. Notice that the **RerouteConnections** method makes it irrelevant what values you supply for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 100, 100).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    .Parent.RerouteConnections
End With
```

EndConnected Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndConnectedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndConnectedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndConnectedA"}

True if the end of the specified connector is connected to a shape. Read-only **Long**.

EndConnected Property Example

If the end of the connector represented by shape three on `myDocument` is connected to a shape, this example stores the connection site number in the variable `oldEndConnSite`, stores a reference to the connected shape in the object variable `oldEndConnShape`, and then disconnects the end of the connector from the shape.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Connector Then
        With .ConnectorFormat
            If .EndConnected Then
                oldEndConnSite = .EndConnectionSite
                Set oldEndConnShape = .EndConnectedShape
                .EndDisconnect
            End If
        End With
    End If
End With
```

EndConnectedShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndConnectedShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndConnectedShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndConnectedShapeA"}

Returns a **Shape** object that represents the shape that the end of the specified connector is attached to. Read-only.

Note If the end of the specified connector isn't attached to a shape, this property generates an error.

EndConnectedShape Property Example

This example assumes that `myDocument` already contains two shapes attached by a connector named "Conn1To2." The code adds a rectangle and a connector to `myDocument`. The end of the new connector will be attached to the same connection site as the end of the connector named "Conn1To2," and the beginning of the new connector will be attached to connection site one on the new rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    Set r3 = .AddShape(msoShapeRectangle, 100, 420, 200, 100)
    .AddConnector(msoConnectorCurve, 0, 0, 10, 10).Name = "Conn1To3"
    With .Item("Conn1To2").ConnectorFormat
        endConnSite1 = .EndConnectionSite
        Set endConnShape1 = .EndConnectedShape
    End With
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect r3, 1
        .EndConnect endConnShape1, endConnSite1
    End With
End With
```

EndConnectionSite Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndConnectionSiteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndConnectionSiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndConnectionSiteA"}

Returns an integer that specifies the connection site that the end of a connector is connected to. Read-only **Long**.

Note If the end of the specified connector isn't attached to a shape, this property generates an error.

EndConnectionSite Property Example

This example assumes that `myDocument` already contains two shapes attached by a connector named "Conn1To2." The code adds a rectangle and a connector to `myDocument`. The end of the new connector will be attached to the same connection site as the end of the connector named "Conn1To2," and the beginning of the new connector will be attached to connection site one on the new rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    Set r3 = .AddShape(msoShapeRectangle, 100, 420, 200, 100)
    .AddConnector(msoConnectorCurve, 0, 0, 10, 10).Name = "Conn1To3"
    With .Item("Conn1To2").ConnectorFormat
        endConnSite1 = .EndConnectionSite
        Set endConnShape1 = .EndConnectedShape
    End With
    With .Item("Conn1To3").ConnectorFormat
        .BeginConnect r3, 1
        .EndConnect endConnShape1, endConnSite1
    End With
End With
```

EndDisconnect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthEndDisconnectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthEndDisconnectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthEndDisconnectA"}

Detaches the end of the specified connector from the shape it's attached to. This method doesn't alter the size or position of the connector: the end of the connector remains positioned at a connection site but is no longer connected. Use the **BeginDisconnect** method to detach the beginning of the connector from a shape.

Syntax

expression.**EndDisconnect**

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

EndDisconnect Method Example

This example adds two rectangles to `myDocument`, attaches them with a connector, automatically reroutes the connector along the shortest path, and then detaches the connector from the rectangles.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 0, 0).ConnectorFormat
    .BeginConnect firstRect, 1
    .EndConnect secondRect, 1
    .Parent.RerouteConnections
    .BeginDisconnect
    .EndDisconnect
End With
```

Activate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthActivateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthActivateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthActivateA"}

Activates the specified object.

Syntax

expression.**Activate**

expression Required. An expression that returns a **DocumentWindow**, **OLEFormat**, or **SlideShowWindow** object.

Activate Method Example

This example activates the document window immediately following the active window in the document window order.

```
Application.ActiveWindow.Next.Activate
```

Active Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActiveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActiveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActiveA"}

True if the specified window is the active window. Read-only **Long**.

Active Property Example

This example activates window three. If window three isn't the active window, the example saves the active window in the variable `oldWin` before activating window three.

```
With Application.Presentations("test.ppt").Windows(1)
    If Not .Active Then
        Set oldWin = Application.ActiveWindow
        .Activate
    End If
End With
```

BlackAndWhite Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproBlackAndWhiteC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproBlackAndWhiteX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproBlackAndWhiteA"}

True if the document window display is black and white. The default value is **False**. Read/write **Long**.

BlackAndWhite Property Example

This example changes the display in window one to black and white.

```
Application.Windows(1).BlackAndWhite = True
```

Close Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCloseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCloseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCloseA"}

Closes the specified document window, presentation, or open freeform drawing.

Syntax

expression.**Close**

expression Required. An expression that returns a **DocumentWindow** or **Presentation** object.

Close Method Example

This example closes all windows except the active window.

```
With Application.Windows
    For i = 2 To .Count
        .Item(2).Close
    Next
End With
```

This example closes Pres1.ppt without saving changes.

```
With Application.Presentations("pres1.ppt")
    .Saved = True
    .Close
End With
```

This example closes all open presentations. If there are changes in any presentation, PowerPoint displays a message asking whether the user wants to save changes.

```
With Application.Presentations
    For i = .Count To 1 Step -1
        .Item(i).Close
    Next
End With
```

FitToPage Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFitToPageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFitToPageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFitToPageA"}

Adjusts the size of the specified document window to accommodate the information that's currently displayed.

Syntax

expression.**FitToPage**

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

FitToPage Method Example

This example sets the view in the active window to slide view, sets the zoom to 25 percent, and adjusts the size of the window to fit the slide displayed there.

```
With Application.ActiveWindow
    .ViewType = ppViewSlide
    .View.Zoom = 25
    .FitToPage
End With
```

IsFullScreen Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproIsFullScreenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproIsFullScreenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproIsFullScreenA"}

True if the specified slide show window occupies the full screen. Read-only **Long**.

IsFullScreen Property Example

This example reduces the height of a full-screen slide show window just enough so that you can see the taskbar.

```
With Application.SlideShowWindows(1)
    If .IsFullScreen Then
        .Height = .Height - 20
    End If
End With
```

LargeScroll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthLargeScrollC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthLargeScrollX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthLargeScrollA"}

Scrolls through the specified document window by pages.

Syntax

expression.**LargeScroll(*Down, Up, ToRight, ToLeft*)**

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

Down Optional **Long**. Specifies the number of pages to scroll down.

Up Optional **Long**. Specifies the number of pages to scroll up.

ToRight Optional **Long**. Specifies the number of pages to scroll right.

ToLeft Optional **Long**. Specifies the number of pages to scroll left.

Remarks

If no arguments are specified, this method scrolls down one page. If ***Down*** and ***Up*** are both specified, their effects are combined. For example, if ***Down*** is 2 and ***Up*** is 4, this method scrolls up two pages. Similarly, if ***Right*** and ***Left*** are both specified, their effects are combined.

Any of the arguments can be a negative number.

LargeScroll Method Example

This example scrolls the active window down three pages.

```
Application.ActiveWindow.LargeScroll Down:=3
```

NewWindow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthNewWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthNewWindowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthNewWindowA"}

Presentation object: Opens a new window that contains the specified presentation. Returns a **DocumentWindow** object that represents the new window.

DocumentWindow object: Opens a new window that contains the same document that's displayed in the specified window. Returns a **DocumentWindow** object that represents the new window.

Syntax

expression.**NewWindow**

expression Required. An expression that returns a **DocumentWindow** or **Presentation** object.

NewWindow Method Example

This example creates a new window with the contents of the active window (this activates the new window) and then switches back to the first window.

```
Set oldW = Application.ActiveWindow
Set newW = oldW.NewWindow
oldW.Activate
```

Next Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthNextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthNextA"}

DocumentWindow object: Returns a **DocumentWindow** object that represents the next window in the document window order. If the **Next** method is applied to the last window in the order, it returns **Nothing**.

SlideShowWindow object: Returns a **SlideShowWindow** object that represents the next window in the slide show window order. If the **Next** method is applied to the last window in the order, it returns **Nothing**.

SlideShowView object: Displays the slide immediately following the slide that's currently displayed. If the last slide is displayed, closes the slide show in speaker mode and returns to the first slide in kiosk mode.

Syntax

expression.**Next**

expression Required. An expression that returns a **DocumentWindow**, **SlideShowView**, or **SlideShowWindow** object.

Next Method Example

This example activates the document window immediately following the active window in the document window order.

```
Application.ActiveWindow.Next.Activate
```

This example shows the slide immediately following the currently displayed slide on slide show window one.

```
Application.SlideShowWindows(1).View.Next
```

Previous Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPreviousC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPreviousX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPreviousA"}

DocumentWindow object: Returns a **DocumentWindow** object that represents the previous window in the document window order. If the **Previous** method is applied to the first window in the order, it returns **Nothing**.

SlideShowWindow object: Returns a **SlideShowWindow** object that represents the previous window in the slide show window order. If the **Previous** method is applied to the first window in the order, it returns **Nothing**.

SlideShowView object: Show the slide immediately preceding the slide that's currently displayed. Has no effect if the first slide in the presentation is currently displayed.

Syntax

expression.**Previous**

expression Required. An expression that returns a **DocumentWindow**, **SlideShowView**, or **SlideShowWindow** object.

Previous Method Example

This example shows the slide immediately preceding the currently displayed slide on slide show window one.

```
Application.SlideShowWindows(1).View.Previous
```

Selection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproSelectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproSelectionA"}

Returns a **Selection** object that represents the selection in the specified document window. Read-only.

Selection Property Example

If there's text selected in the active window, this example makes the text italic.

```
With Application.ActiveWindow.Selection
    If .Type = ppSelectionText Then
        .TextRange.Font.Italic = True
    End If
End With
```

SmallScroll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSmallScrollC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSmallScrollX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSmallScrollA"}

Scrolls through the specified document window by lines and columns.

Syntax

expression.**SmallScroll(*Down, Up, ToRight, ToLeft*)**

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

Down Optional **Long**. Specifies the number of lines to scroll down.

Up Optional **Long**. Specifies the number of lines to scroll up.

ToRight Optional **Long**. Specifies the number of columns to scroll right.

ToLeft Optional **Long**. Specifies the number of columns to scroll left.

Remarks

If no arguments are specified, this method scrolls down one line. If ***Down*** and ***Up*** are both specified, their effects are combined. For example, if ***Down*** is 2 and ***Up*** is 4, this method scrolls up two lines. Similarly, if ***Right*** and ***Left*** are both specified, their effects are combined.

Any of the arguments can be a negative number.

SmallScroll Method Example

This example scrolls down three lines in the active window.

```
Application.ActiveWindow.SmallScroll Down:=3
```

View Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproViewC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproViewX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproViewA"}

DocumentWindow object: Returns a **View** object that represents the view in the specified document window. Read-only.

SlideShowWindow object: Returns a **SlideShowView** object. Read-only.

View Property Example

This example sets the view in the active window to slide view and then displays slide three.

```
With Application.ActiveWindow
    .ViewType = ppViewSlide
    .View.GotoSlide 3
End With
```

ViewType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproViewTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproViewTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproViewTypeA"}

Returns or sets the type of the view contained in the specified document window. Can be one of the following **PpViewType** constants: **ppViewHandout**, **ppViewHandoutMaster**, **ppViewNotesMaster**, **ppViewNotesPage**, **ppViewOutline**, **ppViewSlide**, **ppViewSlideMaster**, **ppViewSlideShow**, **ppViewSlideSorter**, or **ppViewTitleMaster**. Read/write **Long**.

ViewType Property Example

This example changes the view in the active window to slide sorter view if the window is currently displayed in slide view.

```
With Application.ActiveWindow
    If .ViewType = ppViewSlide Then .ViewType = ppViewSlideSorter
End With
```

WindowState Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproWindowStateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproWindowStateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproWindowStateA"}

Returns or sets the state of the specified window. Can be one of the following **PpWindowState** constants: **ppWindowMaximized**, **ppWindowMinimized**, **ppWindowMixed**, or **ppWindowNormal**. Read/write **Long**.

Remarks

When the state of the window is **ppWindowNormal**, the window is neither maximized nor minimized.

WindowState Property Example

This example maximizes the active window.

```
Application.ActiveWindow.WindowState = ppWindowMaximized
```

Arrange Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthArrangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthArrangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthArrangeA"}

Arranges all open document windows in the workspace.

Syntax

expression.**Arrange(*ArrangeStyle*)**

expression An expression that returns a **DocumentWindows** collection.

ArrangeStyle Optional **Long**. Specifies whether to cascade or tile the windows. Can be either of the following **PpArrangeStyle** constants: **ppArrangeCascade**, or **ppArrangeTiled**. The default value is **ppTiled**.

Arrange Method Example

This example creates a new window and then arranges all open document windows.

```
Application.ActiveWindow.NewWindow
Windows.Arrange ppArrangeCascade
```

BackColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBackColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBackColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBackColorA"}

Returns or sets a **ColorFormat** object that represents the background color for the specified fill or patterned line. Read/write.

BackColor Property 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 = ActivePresentation.Slides(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
```

This example adds a patterned line to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

Background Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthBackgroundA"}

Specifies that the shape's fill should match the slide background. If you change the slide background after applying this method to a fill, the fill will also change.

Remarks

Note that applying the **Background** method to a shape's fill isn't the same as setting a transparent fill for the shape, nor is it always the same as applying the same fill to the shape as you apply to the background. The second example demonstrates this.

Background Method Example

This example sets the fill of shape one on slide one in the active presentation to match the slide background.

```
ActivePresentation.Slides(1).Shapes(1).Fill.Background
```

This example sets the background for slide one in the active presentation to a preset gradient, adds a rectangle to the slide, and then places three ovals in front of the rectangle. The first oval has a fill that matches the slide background, the second has a transparent fill, and the third has the same fill applied to it as was applied to the background. Notice the difference in the appearances of these three ovals.

```
With ActivePresentation.Slides(1)
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientDaybreak
    With .Shapes
        .AddShape msoShapeRectangle, 50, 200, 600, 100
        .AddShape(msoShapeOval, 75, 150, 150, 100).Fill.Background
        .AddShape(msoShapeOval, 275, 150, 150, 100).Fill _
            .Transparency = 1
        .AddShape(msoShapeOval, 475, 150, 150, 100).Fill.PresetGradient _
             msoGradientHorizontal, 1, msoGradientDaybreak
    End With
End With
```

ForeColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproForeColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproForeColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproForeColorA"}

Returns or sets a **ColorFormat** object that represents the foreground color for the fill, line, or shadow. Read/write.

ForeColor Property 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 = ActivePresentation.Slides(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
```

This example adds a patterned line to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

GradientColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGradientColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGradientColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGradientColorTypeA"}

Returns the gradient color type for the specified fill. Can be one of the following **MsoGradientColorType** constants: **msoGradientColorMixed**, **msoGradientOneColor**, **msoGradientPresetColors**, or **msoGradientTwoColors**. Read-only **Long**.

This property is read-only. Use the **OneColorGradient**, **PresetGradient**, or **TwoColorGradient** method to set the gradient type for the fill.

GradientColorType Property Example

This example changes the fill for all shapes in `myDocument` that have a two-color gradient fill to a preset gradient fill.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    With s.Fill
        If .GradientColorType = msoGradientTwoColors Then
            .PresetGradient msoGradientHorizontal, 1, msoGradientBrass
        End If
    End With
Next
```

GradientDegree Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGradientDegreeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGradientDegreeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGradientDegreeA"}

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 coloris mixed in. Read-only **Single**.

This property is read-only. Use the **OneColorGradient** method to set the gradient degree for the fill.

GradientDegree Property Example

This example adds a rectangle to `myDocument` 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    gradDegree1 = .Item("Rectangle 2").Fill.GradientDegree
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient msoGradientHorizontal, 1, gradDegree1
    End With
End With
```

GradientStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGradientStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGradientStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGradientStyleA"}

Returns the gradient style for the specified fill. Can be one of the following **MsoGradientStyle** constants: **msoGradientDiagonalDown**, **msoGradientDiagonalUp**, **msoGradientFromCenter**, **msoGradientFromCorner**, **msoGradientFromTitle**, **msoGradientHorizontal**, **msoGradientMixed**, or **msoGradientVertical**. Read-only **Long**.

This property is read-only. Use the **OneColorGradient**, **PresetGradient**, 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.

GradientStyle Property Example

This example adds a rectangle to `myDocument` 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    gradStyle1 = .Item("rect1").Fill.GradientStyle
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient gradStyle1, 1, 1
    End With
End With
```

GradientVariant Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGradientVariantC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGradientVariantX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGradientVariantA"}

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 **msoGradientFromTitle** or **msoGradientFromCenter**, this property returns either 1 or 2. The values for this property correspond to the gradient variants (numbered from left to right and from top to bottom) on the **Gradient** tab in the **Fill Effects** dialog box. Read-only **Long**.

This property is read-only. Use the **OneColorGradient**, **PresetGradient**, or **TwoColorGradient** method to set the gradient variant for the fill.

GradientVariant Property Example

This example adds a rectangle to `myDocument` 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    gradVar1 = .Item("rect1").Fill.GradientVariant
    With .AddShape(msoShapeRectangle, 0, 0, 40, 80).Fill
        .ForeColor.RGB = RGB(128, 0, 0)
        .OneColorGradient msoGradientHorizontal, gradVar1, 1
    End With
End With
```

OneColorGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthOneColorGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthOneColorGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthOneColorGradientA"}

Sets the specified fill to a one-color gradient.

Syntax

expression.**OneColorGradient(*Style, Variant, Degree*)**

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

Style Required **Long**. The gradient style. Can be one of the following **MsoGradientStyle** constants: **msoGradientDiagonalDown**, **msoGradientDiagonalUp**, **msoGradientFromCenter**, **msoGradientFromCorner**, **msoGradientFromTitle**, **msoGradientHorizontal**, or **msoGradientVertical**.

Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

Degree Required **Single**. The gradient degree. Can be a value from 0.0 (dark) to 1.0 (light).

OneColorGradient Method Example

This example adds a rectangle with a one-color gradient fill to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 90, 90, 90, 80).Fill
    .ForeColor.RGB = RGB(0, 128, 128)
    .OneColorGradient msoGradientHorizontal, 1, 1
End With
```

Pattern Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPatternC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPatternX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPatternA"}

Returns a value that represents the pattern applied to the specified fill or line. Read-only **Long**.

Can be one of the following **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

This property is read-only. 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.

Pattern Property 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 = ActivePresentation.Slides(1)
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 = ActivePresentation.Slides(1)
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
```

Patterned Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPatternedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPatternedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPatternedA"}

Sets the specified fill to a pattern.

Syntax

expression.**Patterned(*Pattern*)**

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

Pattern Required **Long**. The pattern to be used for the specified fill. Can be any of the **MsoPatternType** constants.

Remarks

Use the **BackColor** and **ForeColor** properties to set the colors used in the pattern.

Patterned Method Example

This example adds an oval with a patterned fill to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeOval, 60, 60, 80, 40).Fill
    .ForeColor.RGB = RGB(128, 0, 0)
    .BackColor.RGB = RGB(0, 0, 255)
    .Patterned msoPatternDarkVertical
End With
```

PresetGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPresetGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPresetGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPresetGradientA"}

Sets the specified fill to a preset gradient.

Syntax

expression.**PresetGradient(*Style, Variant, PresetGradientType*)**

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

Style Required **Long**. The gradient style. Can be one of the following **MsoGradientStyle** constants: **msoGradientDiagonalDown**, **msoGradientDiagonalUp**, **msoGradientFromCenter**, **msoGradientFromCorner**, **msoGradientFromTitle**, **msoGradientHorizontal**, or **msoGradientVertical**.

Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

PresetGradientType Required **Long**. The gradient type. Can be one of the following **MsoPresetGradientType** constants:

msoGradientBrass	**msoGradientLateSunset**
msoGradientCalmWater	**msoGradientMahogany**
msoGradientChrome	**msoGradientMoss**
msoGradientChromeII	**msoGradientNightfall**
msoGradientDaybreak	**msoGradientOcean**
msoGradientDesert	**msoGradientParchment**
msoGradientEarlySunset	**msoGradientPeacock**
msoGradientFire	**msoGradientRainbow**
msoGradientFog	**msoGradientRainbowII**
msoGradientGold	**msoGradientSapphire**
msoGradientGoldII	**msoGradientSilver**
msoGradientHorizon	**msoGradientWheat**

PresetGradient Method Example

This example adds a rectangle with a preset gradient fill to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape(msoShapeRectangle, 90, 90, 140, 80) _
    .Fill.PresetGradient msoGradientHorizontal, 1, msoGradientBrass
```

PresetGradientType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetGradientTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetGradientTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetGradientTypeA"}

Returns the preset gradient type for the specified fill. Read-only **Long**.

Can be one of the following **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

This property is read-only. Use the **PresetGradient** method to set the preset gradient type for the fill.

PresetGradientType Property 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 = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    With s.Fill
        If .PresetGradientType = msoGradientMoss Then
            .PresetGradient msoGradientHorizontal, 1, msoGradientFog
        End If
    End With
Next
```

PresetTexture Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPresetTextureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPresetTextureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPresetTextureA"}

Returns the preset texture for the specified fill. Read-only **Long**.

Can be one of the following **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

This property is read-only. Use the **PresetTextured** method to set the preset texture for the fill.

PresetTexture Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes
    presetTexture2 = .Item(2).Fill.PresetTexture
    .AddShape(msoShapeRectangle, 100, 0, 40, 80).Fill _
        .PresetTextured presetTexture2
End With
```

PresetTextured Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPresetTexturedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPresetTexturedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPresetTexturedA"}

Sets the specified fill to a preset texture.

Syntax

expression.**PresetTextured(*PresetTexture*)**

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

PresetTexture Required **Long**. The preset texture. Can be one of the following **MsoPresetTexture** constants:

msoTextureBlueTissuePaper	**msoTexturePaperBag**
msoTextureBouquet	**msoTexturePapyrus**
msoTextureBrownMarble	**msoTextureParchment**
msoTextureCanvas	**msoTexturePinkTissuePaper**
msoTextureCork	**msoTexturePurpleMesh**
msoTextureDenim	**msoTextureRecycledPaper**
msoTextureFishFossil	**msoTextureSand**
msoTextureGranite	**msoTextureStationery**
msoTextureGreenMarble	**msoTextureWalnut**
msoTextureMediumWood	**msoTextureWaterDroplets**
msoTextureNewsprint	**msoTextureWhiteMarble**
msoTextureOak	**msoTextureWovenMat**

PresetTextured Method Example

This example adds a rectangle with a green-marble textured fill to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape(msoShapeCan, 90, 90, 40, 80) _
    .Fill.PresetTextured msoTextureGreenMarble
```

Solid Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSolidC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSolidX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSolidA"}

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.

Syntax

expression.**Solid**

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

Solid Method Example

This example converts all fills on `myDocument` to uniform red fills.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    With s.Fill
        .Solid
        .ForeColor.RGB = RGB(255, 0, 0)
    End With
Next
```

TextureName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextureNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextureNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextureNameA"}

Returns the name of the custom texture file for the specified fill. Read-only **String**.

This property is read-only. Use the **UserTextured** method to set the texture file for the fill.

TextureName Property Example

This example adds an oval to `myDocument`. If shape one on `myDocument` has a user-defined textured fill, the new oval will have the same fill as shape one. If shape one has any other type of fill, the new oval will have a green marble fill.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    Set newFill = .AddShape(msoShapeOval, 0, 0, 200, 90).Fill
    With .Item(1).Fill
        If .Type = msoFillTextured And _
                .TextureType = msoTextureUserDefined Then
            newFill.UserTextured .TextureName
        Else
            newFill.PresetTextured msoTextureGreenMarble
        End If
    End With
End With
```

TextureType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextureTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextureTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextureTypeA"}

Returns the texture type for the specified fill. Can be one of the following **MsoTextureType** constants: **msoTexturePreset**, **msoTextureTypeMixed**, or **msoTextureUserDefined**. Read-only **Long**.

This property is read-only. Use the **PresetTextured** or **UserTextured** method to set the texture type for the fill.

TextureType Property Example

This example changes the fill for all shapes on `myDocument` with a custom textured fill to a canvas fill.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    With s.Fill
        If .TextureType = msoTextureUserDefined Then
            .PresetTextured msoTextureCanvas
        End If
    End With
Next
```

TwoColorGradient Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthTwoColorGradientC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthTwoColorGradientX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthTwoColorGradientA"}

Sets the specified fill to a two-color gradient.

Syntax

expression.**TwoColorGradient(*Style, Variant*)**

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

Style Required **Long**. The gradient style. Can be one of the following **MsoGradientStyle** constants: **msoGradientDiagonalDown**, **msoGradientDiagonalUp**, **msoGradientFromCenter**, **msoGradientFromCorner**, **msoGradientFromTitle**, **msoGradientHorizontal**, or **msoGradientVertical**.

Variant Required **Long**. The gradient variant. Can be a value from 1 to 4, corresponding to the four variants on the **Gradient** tab in the **Fill Effects** dialog box. If ***Style*** is **msoGradientFromTitle** or **msoGradientFromCenter**, this argument can be either 1 or 2.

TwoColorGradient Method Example

This example adds a rectangle with a two-color gradient fill to `myDocument` and sets the background and foreground color for the fill.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

UserPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUserPictureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUserPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUserPictureA"}

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.

Syntax

expression.**UserPicture(*PictureFile*)**

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

PictureFile Required **String**. The name of the picture file.

UserPicture Method Example

This example adds two rectangles to `myDocument`. 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

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

UserTextured Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUserTexturedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUserTexturedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUserTexturedA"}

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.

Syntax

expression.**UserTextured(*TextureFile*)**

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

TextureFile Required **String**. The name of the picture file.

UserTextured Method Example

This example adds two rectangles to `myDocument`. 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

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

BaselineOffset Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproBaselineOffsetC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproBaselineOffsetX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproBaselineOffsetA"}

Returns or sets the baseline offset for the specified superscript or subscript characters. Can be a floating-point value from – 1 through 1. A value of – 1 represents an offset of – 100 percent, and a value of 1 represents an offset of 100 percent. Read/write **Single**.

Remarks

Setting the **BaselineOffset** property to a negative value automatically sets the **Subscript** property to **True** and the **Superscript** property to **False**.

Setting the **BaselineOffset** property to a positive value automatically sets the **Subscript** property to **False** and the **Superscript** property to **True**.

Setting the **Subscript** property to **True** automatically sets the **BaselineOffset** property to 0.3 (30 percent).

Setting the **Superscript** property to **True** automatically sets the **BaselineOffset** property to – 0.25 (– 25 percent).

BaselineOffset Property Example

This example sets the text for shape two on slide one and then makes the second character subscript with a 20-percent offset.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange
    .Text = "H2O"
    .Characters(2, 1).Font.BaselineOffset = -0.2
End With
```

Bold Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBoldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBoldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBoldA"}

True if the character format is bold, **False** if it's not bold, or **msoTriStateMixed** if the specified text range contains both bold and nonbold characters. Read/write **Long**.

Bold Property Example

This example sets characters one through five in the title on slide one to bold.

```
Set myT = Application.ActivePresentation.Slides(1).Shapes.Title
myT.TextFrame.TextRange.Characters(1, 5).Font.Bold = True
```

Color Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproColorA"}

Returns a **ColorFormat** object that represents the background color for the specified characters. Read-only.

Color Property Example

This example sets the color of characters one through five in the title on slide one.

```
myRed = RGB(255, 0, 0)
Set myT = Application.ActivePresentation.Slides(1).Shapes.Title
myT.TextFrame.TextRange.Characters(1, 5).Font.Color.RGB = myRed
```

Emboss Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEmbossC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEmbossX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEmbossA"}

True if the character format is embossed, **False** if it's not embossed, or **msoTriStateMixed** if the specified text range contains both embossed and unembossed characters. Read/write **Long**.

Emboss Property Example

This example sets the title text on slide one to embossed.

```
Application.ActivePresentation.Slides(1).Shapes.Title _
    .TextFrame.TextRange.Font.Emboss = True
```

Italic Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproItalicC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproItalicX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproItalicA"}

Font object: **True** if the character format is italic, **False** if it's not italic, or **msoTriStateMixed** if the specified text range contains both italic and nonitalic characters. Read/write **Long**.

Italic Property Example

This example sets the title text on slide one and makes the title blue and italic.

```
With
Application.ActivePresentation.Slides(1).Shapes.Title.TextFrame.TextRange
    .Text = "Volcano Coffee"
    With .Font
        .Italic = True
        .Name = "palatino"
        .Color.RGB = RGB(0, 0, 255)
    End With
End With
```

Shadow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShadowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShadowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShadowA"}

Font object: **True** if the specified text has a shadow, **False** if it doesn't have a shadow, or **msoTriStateMixed** if some of the text has a shadow and some doesn't. Read/write **Long**.

Shape or **ShapeRange** object: Returns a **ShadowFormat** object that contains shadow formatting properties for the specified shape or shapes. Read-only.

Shadow Property Example

This example adds a shadow to the title text on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes.Title _
    .TextFrame.TextRange.Font.Shadow = True
```

This example adds a shadowed rectangle to slide one in the active presentation. The blue, embossed shadow is offset 3 points to the right of and 2 points down from the rectangle.

```
Set myShap = Application.ActivePresentation.Slides(1).Shapes
With myShap.AddShape(msoShapeRectangle, 10, 10, 150, 90).Shadow
    .Type = msoShadow17
    .ForeColor.RGB = RGB(0, 0, 128)
    .OffsetX = 3
    .OffsetY = 2
End With
```

Size Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSizeA"}

Returns or sets the character size, in points. Read/write **Single**.

Size Property Example

This example sets the size of the text attached to shape one on slide one to 24 points.

```
Application.ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange.Font
. _
    Size = 24
```

Subscript Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSubscriptC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSubscriptX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSubscriptA"}

True if the specified text is subscript, **False** if it's not subscript, or **msoStateMixed** if some characters are subscript and some aren't. The default value is **False**. Read/write **Long**.

Remarks

Setting the **BaselineOffset** property to a negative value automatically sets the **Subscript** property to **True** and the **Superscript** property to **False**.

Setting the **BaselineOffset** property to a positive value automatically sets the **Subscript** property to **False** and the **Superscript** property to **True**.

Setting the **Subscript** property to **True** automatically sets the **BaselineOffset** property to – 0.25 (– 25 percent).

Subscript Property Example

This example enlarges the first character in the title on slide one if that character is subscript.

```
With
Application.ActivePresentation.Slides(1).Shapes.Title.TextFrame.TextRange
    With .Characters(1, 1).Font
        If .Subscript Then
            scaleChar = -20 * .BaselineOffset
            .Size = .Size * scaleChar
        End If
    End With
End With
```

Superscript Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproSuperscriptC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproSuperscriptX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproSuperscriptA"}

True if the specified text is superscript, **False** if it's not superscript, or **msoStateMixed** if some characters are superscript and some aren't. The default value is **False**. Read/write **Long**.

Remarks

Setting the **BaselineOffset** property to a negative value automatically sets the **Subscript** property to **True** and the **Superscript** property to **False**.

Setting the **BaselineOffset** property to a positive value automatically sets the **Subscript** property to **False** and the **Superscript** property to **True**.

Setting the **Superscript** property to **True** automatically sets the **BaselineOffset** property to 0.3 (30 percent).

Superscript Property Example

This example sets the text for shape two on slide one and then makes the fifth character superscript with a 30-percent offset.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame
    With .TextRange
        .Text = "E=mc2"
        .Characters(5, 1).Font.Superscript = True
    End With
End With
```

Underline Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproUnderlineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproUnderlineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproUnderlineA"}

Font object: **True** if the specified text is underlined, **False** if it's not underlined, or **msoTriStateMixed** if some characters are underlined and some aren't. Read/write **Long**.

FontInfo object: **True** if the font style is underlined. Read-only **Long**.

Underline Property Example

This example sets the formatting for the text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.Font
        .Size = 32
        .Name = "Palatino"
        .Underline = True
    End With
End With
```

Embeddable Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEmbeddableC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEmbeddableX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEmbeddableA"}

True if the specified font can be embedded in the presentation. Read-only **Long**.

Embeddable Property Example

This example checks each font used in the active presentation to determine whether it's embeddable in the presentation.

```
For Each usedFont In Presentations(1).Fonts
    If usedFont.Embeddable Then
        MsgBox usedFont.Name & ": Embeddable"
    Else
        MsgBox usedFont.Name & ": Not embeddable"
    End If
Next usedFont
```

Embedded Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEmbeddedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEmbeddedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEmbeddedA"}

True if the specified font is embedded in the presentation. Read-only **Long**.

Embedded Property Example

This example checks each font used in the active presentation to determine whether it's embedded in the presentation.

```
For Each usedFont In Presentations(1).Fonts
    If usedFont.Embedded Then
        MsgBox usedFont.Name & ": Embedded"
    Else
        MsgBox usedFont.Name & ": Not embedded"
    End If
Next usedFont
```

AddNodes Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddNodesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddNodesA"}

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.

Syntax

expression.**AddNodes(*SegmentType, EditingType, X1, Y1, X2, Y2, X3, Y3*)**

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

SegmentType Required **Long**. The type of segment to be added. Can be either of the following **MsoSegmentType** constants: **msoSegmentCurve** or **msoSegmentLine**.

EditingType Required **Long**. The editing property of the vertex. Can be either of the following **MsoEditingType** constants: **msoEditingAuto** or **msoEditingCorner** (cannot be **msoEditingSmooth** or **msoEditingSymmetric**). If ***SegmentType*** is **msoSegmentLine**, ***EditingType*** must be **msoEditingAuto**.

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.

AddNodes Method Example

This example adds a freeform with five vertices to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

ConvertToShape Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthConvertToShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthConvertToShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthConvertToShapeA"}

Creates a shape that has the geometric characteristics of the specified **FreeformBuilder** object. Returns a **Shape** object that represents the new shape.

Note You must apply the **AddNodes** method to a **FreeformBuilder** object at least once before you use the **ConvertToShape** method.

Syntax

expression.**ConvertToShape**

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

ConvertToShape Method Example

This example adds a freeform with five vertices to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

Format Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFormatA"}

Returns or sets the format for the automatically updated date and time. Applies only to **HeaderFooter** objects that represent a date and time (returned from the **HeadersFooters** collection by the **DateAndTime** property). Read/write **Long**.

Can be one of the following **PpDateTimeFormat** constants:

ppDateTimeddddMMMMddyyyy	**ppDateTimehmmssAMPM**
ppDateTimedMMMMyyyy	**ppDateTimeMdyy**
ppDateTimedMMMyy	**ppDateTimeMMddyyHmm**
ppDateTimeFormatMixed	**ppDateTimeMMddyyhmmAMPM**
ppDateTimeHmm	**ppDateTimeMMMMdyyyy**
ppDateTimehmmAMPM	**ppDateTimeMMMMyy**
ppDateTimeHmmss	**ppDateTimeMMyy**

Remarks

Make sure that the date and time are set to be updated automatically (not displayed as fixed text) by setting the **UseFormat** property to **True**.

Format Property Example

This example sets the date and time for the slide master of the active presentation to be updated automatically and then it sets the date and time format to show hours, minutes, and seconds.

```
Set myPres = Application.ActivePresentation
With myPres.SlideMaster.HeadersFooters.DateAndTime
    .UseFormat = True
    .Format = ppDateTimeHmmss
End With
```

Text Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextA"}

HeaderFooter, **TextEffectFormat**, or **TextRange** object: Returns or sets the text contained in the specified object. Read/write **String**.

Text Property Example

This example sets the text and font style for the title on slide one in the active presentation.

```
Set myPres = Application.ActivePresentation
With myPres.Slides(1).Shapes.Title.TextFrame.TextRange
    .Text = "Welcome!"
    .Font.Italic = True
End With
```

UseFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproUseFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproUseFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproUseFormatA"}

True if the date and time object contains automatically updated information; **False** if the date and time is a fixed string. Read/write **Long**.

Remarks

This property applies only to a **HeaderFooter** object that represents a date and time (returned by the **DateAndTime** property). Set the **UseFormat** property of a date and time **HeaderFooter** object to **True** when you want to set or return the date and time format by using the **Format** property. Set the **UseFormat** property to **False** when you want to set or return the text string for the fixed date and time.

UseFormat Property Example

This example sets the date and time for the slide master of the active presentation to be updated automatically and then it sets the date and time format to show hours, minutes, and seconds.

```
Set myPres = Application.ActivePresentation
With myPres.SlideMaster.HeadersFooters.DateAndTime
    .UseFormat = True
    .Format = ppDateTimeHmmss
End With
```

DateAndTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDateAndTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDateAndTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDateAndTimeA"}

Returns a **HeaderFooter** object that represents the date and time item that appears in the lower-left corner of a slide or in the upper-right corner of a notes page, handout, or outline. Read-only.

DateAndTime Property Example

This example sets the date and time format for the slide master in the active presentation. This setting will apply to all slides that are based on this master.

```
Set myPres = Application.ActivePresentation
With myPres.SlideMaster.HeadersFooters.DateAndTime
    .Format = ppDateTimeMdyy
    .UseFormat = True
End With
```

DisplayOnTitleSlide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDisplayOnTitleSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDisplayOnTitleSlideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDisplayOnTitleSlideA"}

True if the footer, date and time, and slide number appear on the title slide. **False** if this information appears on all slides except the title slide. Applies to slide masters. Read/write **Long**.

DisplayOnTitleSlide Property Example

This example sets the footer, date and time, and slide number to not appear on the title slide.

```
Application.ActivePresentation.SlideMaster.HeadersFooters _
    .DisplayOnTitleSlide = False
```

Footer Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFooterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFooterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFooterA"}

Returns a **HeaderFooter** object that represents the footer that appears at the bottom of a slide or in the lower-left corner of a notes page, handout, or outline. Read-only.

Footer Property Example

This example sets the text for the footer on slide one in the active presentation and sets the footer, date and time, and slide number to appear on the title slide.

```
With Application.ActivePresentation.Slides(1).HeadersFooters
    .Footer.Text = "Introduction"
    .DisplayOnTitleSlide = True
End With
```

Header Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHeaderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHeaderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHeaderA"}

Returns a **HeaderFooter** object that represents the header that appears at the top of a slide or in the upper-left corner of a notes page, handout, or outline. Read-only.

Header Property Example

This example sets the header text for the handout master for the active presentation. This text will appear in the upper-left corner of the page when you print your presentation as an outline or a handout.

```
Set myHandHF = Application.ActivePresentation.HandoutMaster.HeadersFooters
myHandHF.Header.Text = "Third Quarter Report"
```

SlideNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideNumberC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideNumberA"}

HeadersFooters collection: Returns a **HeaderFooter** object that represents the slide number in the lower-right corner of a slide, or the page number in the lower-right corner of a notes page or a page of a printed handout or outline. Read-only.

Slide or **SlideRange** object: Returns the slide number. Read-only **Integer**.

Remarks

The **SlideNumber** property of a **Slide** object is the actual number that appears in the lower-right corner of the slide when you display slide numbers. This number is determined by the number of the slide within the presentation (the **SlideIndex** property value) and the starting slide number for the presentation (the **FirstSlideNumber** property value). The slide number is always equal to the the starting slide number + the slide index number – 1.

SlideNumber Property Example

This example shows how changing the first slide number affects the slide number of a specific slide.

```
With Application.ActivePresentation
    .PageSetup.FirstSlideNumber = 1  'starts slide numbering at 1
    MsgBox .Slides(2).SlideNumber    'returns 2

    .PageSetup.FirstSlideNumber = 10  'starts slide numbering at 10
    MsgBox .Slides(2).SlideNumber     'returns 11
End With
```

This example hides the slide number on slide two in the active presentation if the number is currently visible, or it displays the slide number if it's currently hidden.

```
With Application.ActivePresentation.Slides(2).HeadersFooters.SlideNumber
    If .Visible Then
        .Visible = False
    Else
        .Visible = True
    End If
End With
```

Export Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthExportC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthExportX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthExportA"}

Presentation object: Exports each slide in the presentation, using the specified graphics filter, and saves the exported files in the specified folder.

Slide or **SlideRange** object: Exports a slide, using the specified graphics filter, and saves the exported file under the specified file name.

Syntax 1

expression.**Export(*Path, FilterName, ScaleWidth, ScaleHeight*)**

Syntax 2

expression.**Export(*FileName, FilterName, ScaleWidth, ScaleHeight*)**

expression Required. An expression that returns a **Presentation** object (Syntax 1) or a **Slide** or **SlideRange** object (Syntax 2).

Path Required **String**. The path of the folder where you want to save the exported slides. You can include a full path; if you don't do this, PowerPoint creates a subfolder in the current folder for the exported slides.

FileName Required **String**. The name of the file to be exported and saved to disk. You can include a full path; if you don't, PowerPoint creates a file in the current folder.

FilterName Required **String**. The graphics format in which you want to export slides. The specified graphics format must have an export filter registered in the system registry. You can specify either the registered extension or the registered filter name. PowerPoint will first search for a matching extension in the registry. If no extension that matches the specified string is found, PowerPoint will look for a filter name that matches.

ScaleWidth Optional **Long**. The factor by which to scale the width of an exported slide.

ScaleHeight Optional **Long**. The factor by which to scale the height of an exported slide.

Remarks

Exporting a presentation doesn't set the **Saved** property of a presentation to **True**.

PowerPoint uses the specified graphics filter to save each individual slide in the presentation. The names of the slides exported and saved to disk are determined by PowerPoint. They're typically saved as Slide1.WMF, Slide2.WMF, and so on (for example). The path of the saved files is specified in the ***Path*** argument.

Export Method Example

This example saves the active presentation as a PowerPoint presentation and then exports each slide in the presentation as a Portable Network Graphics (PNG) file that will be saved in the Current Work folder. The example also scales each exported slide to half the height and width of the original.

```
With ActivePresentation
    .SaveAs "c:\current work\Annual Sales", ppSaveAsPresentation
    .Export "c:\current work", "png", 0.5, 0.5
End With
```

This example exports slide three in the active presentation to disk in the JPEG graphic format. The slide is saved as "Slide 3 of Annual Sales.jpg."

```
With Application.ActivePresentation.Slides(3)
    .Export "c:\my documents\Graphic Format\Slide 3 of Annual Sales", "JPG"
End With
```

LastSlideViewed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLastSlideViewedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLastSlideViewedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLastSlideViewedA"}

Returns a **Slide** object that represents the slide viewed immediately before the current slide in the specified slide show view.

LastSlideViewed Property Example

This example takes you to the slide viewed immediately before the current slide in slide show window one.

```
With SlideShowWindows(1).View
    .GotoSlide .LastSlideViewed.SlideIndex
End With
```

x-, y-, and z-axes

The three mutually perpendicular lines that are used to locate a point in a Cartesian coordinate system. Note that in the Microsoft Office drawing layer, the x-axis runs horizontally across your document, the y-axis runs vertically, and the z-axis runs perpendicular to the plane of your document (sticking out toward you).

Regroup Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRegroupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRegroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRegroupA"}

Regroups the group that the specified shape range belonged to previously. Returns the regrouped shapes as a single **Shape** object.

Syntax

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.

Regroup Method 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.

```
ActiveWindow.Selection.ShapeRange.Regroup
```

CropBottom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCropBottomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCropBottomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCropBottomA"}

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.

CropBottom Property Example

This example crops 20 points off the bottom of shape three on `myDocument`. For the example to work, shape three must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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.

```
percentToCrop = InputBox("What percentage do you want to crop off the
bottom of this picture?")
Set shapeToCrop = ActiveWindow.Selection.ShapeRange(1)
With shapeToCrop.Duplicate
    .ScaleHeight 1, True
    origHeight = .Height
    .Delete
End With
cropPoints = origHeight * percentToCrop / 100
shapeToCrop.PictureFormat.CropBottom = cropPoints
```

CropLeft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCropLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCropLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCropLeftA"}

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.

CropLeft Property Example

This example crops 20 points off the left side of shape three on `myDocument`. For the example to work, shape three must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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.

```
percentToCrop = InputBox("What percentage do you want to crop off the left
of this picture?")
Set shapeToCrop = ActiveWindow.Selection.ShapeRange(1)
With shapeToCrop.Duplicate
    .ScaleWidth 1, True
    origWidth = .Width
    .Delete
End With
cropPoints = origWidth * percentToCrop / 100
shapeToCrop.PictureFormat.CropLeft = cropPoints
```

CropRight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCropRightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCropRightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCropRightA"}

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.

CropRight Property Example

This example crops 20 points off the right side of shape three on `myDocument`. For this example to work, shape three must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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.

```
percentToCrop = InputBox("What percentage do you want to crop off the right
of this picture?")
Set shapeToCrop = ActiveWindow.Selection.ShapeRange(1)
With shapeToCrop.Duplicate
    .ScaleWidth 1, True
    origWidth = .Width
    .Delete
End With
cropPoints = origWidth * percentToCrop / 100
shapeToCrop.PictureFormat.CropRight = cropPoints
```

CropTop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCropTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCropTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCropTopA"}

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.

CropTop Property Example

This example crops 20 points off the top of shape three on `myDocument`. For the example to work, shape three must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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.

```
percentToCrop = InputBox("What percentage do you want to crop off the top
of this picture?")
Set shapeToCrop = ActiveWindow.Selection.ShapeRange(1)
With shapeToCrop.Duplicate
    .ScaleHeight 1, True
    origHeight = .Height
    .Delete
End With
cropPoints = origHeight * percentToCrop / 100
shapeToCrop.PictureFormat.CropTop = cropPoints
```

Distribute Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDistributeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDistributeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDistributeA"}

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 slide or just over the space they originally occupy.

Syntax

expression.**Distribute(*DistributeCmd, RelativeTo*)**

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

DistributeCmd Required **Long**. Specifies whether shapes in the range are to be distributed horizontally or vertically. Can be either of the following **MsoDistributeCmd** constants: **msoDistributeHorizontally** or **msoDistributeVertically**.

RelativeTo Required **Long**. **True** to distribute the shapes evenly over the entire horizontal or vertical space on the slide. **False** to distribute them within the horizontal or vertical space that the range of shapes originally occupies.

Distribute Method Example

This example defines a shape range that contains all the AutoShapes on `myDocument` and then horizontally distributes the shapes in this range.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

Add Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddC"}

Adds an object to a collection. Select one of the following objects to see a detailed description of the **Add** method for that object.

AddIns

ColorSchemes

ExtraColors

NamedSlideShows

Presentations

PrintRanges

Slides

TabStops

Tags

Hyperlink Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHyperlinkC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHyperlinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHyperlinkA"}

Returns a **Hyperlink** object that represents the hyperlink for the specified shape. For the hyperlink to be active during a slide show, the **Action** property must be set to **ppActionHyperlink**. Read-only.

Hyperlink Property Example

This example sets shape one on slide one in the the active presentation to jump to the Microsoft Web site when the shape is clicked during a slide show.

```
With ActivePresentation.Slides(1).Shapes(1).ActionSettings(ppMouseClick)
    .Action = ppActionHyperlink
    .Hyperlink.Address = "http://www.microsoft.com"
End With
```

Address Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAddressC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAddressX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAddressA"}

Returns or sets the address of the target document. Read/write **String**.

Address Property Example

This example scans `myDocument` for a hyperlink to the Microsoft Web site.

```
For Each s In myDocument.Hyperlinks
    If s.Address = "http://www.microsoft.com" Then
        MsgBox "You have a link to the Microsft Home Page"
    End If
Next
```

SubAddress Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSubAddressC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSubAddressX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSubAddressA"}

Returns or sets the location within a document – such as a bookmark in a word document, a range in a Microsoft Excel worksheet, or a slide in a PowerPoint presentation – associated with the specified hyperlink. Read/write **String**.

SubAddress Property Example

This example sets shape one on slide one in the active presentation to jump to the slide named "Last Quarter" in Latest Figures.ppt when the shape is clicked during a slide show.

```
With ActivePresentation.Slides(1).Shapes(1).ActionSettings(ppMouseClick)
    .Action = ppActionHyperlink
    With .Hyperlink
        .Address = = "c:\sales\latest figures.ppt"
        .SubAddress = "last quarter"
    End With
End With
```

This example sets shape one on slide one in the active presentation to jump to range A1:B10 in Latest.xls when the shape is clicked during a slide show.

```
With ActivePresentation.Slides(1).Shapes(1).ActionSettings(ppMouseClick)
    .Action = ppActionHyperlink
    With .Hyperlink
        .Address = = "c:\sales\latest.xls"
        .SubAddress = "A1:B10"
    End With
End With
```

Item Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemC"}

Returns a single member of a collection. Depending on the collection, you use an index number, a name, or a built-in constant to specify which member to return.

Item is the default member for a collection. For example, the following two lines of code are equivalent:

```
ActivePresentation.Slides.Item(1)
ActivePresentation.Slides(1)
```

For more information about returning a single member of a collection, see Returning an Object from a Collection.

For specific information about the **Item** method for a given collection, select the name of the collection in the following list.

ActionSettings
AddIns
ColorSchemes
DocumentWindows
ExtraColors
Fonts
GroupShapes
Hyperlinks
NamedSlideShows
ObjectVerbs
Placeholders
Presentations
PrintRanges
RulerLevels
ShapeNodes
ShapeRange
Shapes
SlideRange
Slides
SlideShowWindows
TabStops
Tags
TextStyleLevels

Remarks

The **Item** method generally returns an object whose name is the singular form of the collection to which the method is applied. For example, the **Item** method of the **Shapes** collection returns a **Shape** object. The exceptions to this rule are listed in the following table.

Object	Return type of the Item method
ExtraColors	**Long**
GroupShapes	**Shape**
ObjectVerbs	**String**
Placeholders	**Shape**
ShapeRange	**Shape**
SlideRange	**Slide**
Tags	**String**

Item Method (Multiple Collections)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemGroupOneObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemGroupOneObjA"}

Returns a single member of a collection.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns an **AddIns**, **Fonts**, **GroupShapes**, **Hyperlinks**, **NamedSlideShows**, **Presentations**, **ShapeNodes**, **ShapeRange**, **Shapes**, **SlideRange**, or **Slides** collection.

Index Required **Variant**. The name or index number of the member in the collection to be returned.

Item Method (Multiple Collections)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemGroupTwoObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemGroupTwoObjA"}

Returns a single member of a collection.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns a **ColorSchemes**, **DocumentWindows**, **Placeholders**, **PrintRanges**, **RulerLevels**, **SlideShowWindows**, or **TabStops** collection.

Index Required **Long**. The index number of the member in the collection to be returned.

Item Method (ActionSettings Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemActionSettingsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemActionSettingsObjA"}

Returns a single member of a collection of action settings.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns an **ActionSettings** collection.

Index Required **Long**. Specifies which of the following two settings is to be returned: the action setting for when the mouse pointer is positioned over the specified shape, or the action setting for when the user clicks the shape. Can be either of the following **PpMouseActivation** constants: **ppMouseClick** or **ppMouseOver**.

Item Method (TextStyleLevels Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemTextStyleLevelsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemTextStyleLevelsObjA"}

Returns a single member of the collection of text style levels.

Syntax

expression.**Item(*Level*)**

expression Required. An expression that returns a **TextStyleLevels** collection.

Level Required **Long**. The level to be returned from the collection. Can be an integer from 1 through 5.

Item Method (TextStyles Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemTextStylesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemTextStylesObjA"}

Returns a single member of the collection of text styles.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns a **TextStyles** collection.

Index Required **Long**. The style to be returned from the collection. Can be one of the following **PpTextStyleType** constants: **ppBodyStyle**, **ppDefaultStyle**, or **ppTitleStyle**.

Item Method (Tags Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemTagsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemTagsObjA"}

Returns a single tag from the **Tags** object.

Syntax

expression.**Item(*Name*)**

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

Name Required **String**. The name of the member in the collection to be returned.

Item Method (ExtraColors Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemExtraColorsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemExtraColorsObjA"}

Returns a **Long** value that represents one of the extra colors in the presentation.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns an **ExtraColors** collection.

Index Required **Long**. The index number of the extra color to return.

Item Method (ObjectVerbs Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthItemObjectVerbsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthItemObjectVerbsObjA"}

Returns a **String** value that represents one of the OLE verbs for the specified OLE object.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns an **ObjectVerbs** collection.

Index Required **Long**. The index number of the verb to return.

Item Method Example

This example sets the foreground color to red for the shape named "Rectangle 1" on slide one in the active presentation.

```
ActivePresentation.Slides.Item(1).Shapes.Item("rectangle 1").Fill _
    .ForeColor.RGB = RGB(128, 0, 0)
```

This example sets shape three on slide one to play the sound of applause when it's clicked during a slide show and specifies that the shape's color be momentarily inverted whenever the mouse pointer passes over the shape during a slide show.

```
With
ActivePresentation.Slides.Item(1).Shapes.Item(3).ActionSettings.Item(ppMous
eClick)
    .SoundEffect.Name = "applause.wav"
    .AnimateAction = True
End With
```

This example sets the first-line indent and the hanging indent for outline level one in body text on the slide master for the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles.Item(ppBodyStyle)
    With .Ruler.Levels.Item(1) ' sets indents for level 1
        .FirstMargin = 9
        .LeftMargin = 54
    End With
End With
```

This example hides all slides in the active presentation that don't have the value "east" for the "region" tag.

```
For Each s In ActivePresentation.Slides
    If s.Tags.Item("region") <> "east" Then
        s.SlideShowTransition.Hidden = True
    End If
Next
```

BeginArrowheadLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BeginArrowheadLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BeginArrowheadLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BeginArrowheadLengthA"}

Returns or sets the length of the arrowhead at the beginning of the specified line. Can be one of the following **MsoArrowheadLength** constants: **msoArrowheadLengthMedium**, **msoArrowheadLengthMixed**, **msoArrowheadLong**, or **msoArrowheadShort**. Read/write **Long**.

BeginArrowheadLength Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

BeginArrowheadStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BeginArrowheadStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BeginArrowheadStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BeginArrowheadStyleA"}

Returns or sets the style of the arrowhead at the beginning of the specified line. Can be one of the following **MsoArrowheadStyle** constants: **msoArrowheadDiamond**, **msoArrowheadNone**, **msoArrowheadOpen**, **msoArrowheadOval**, **msoArrowheadStealth**, **msoArrowheadStyleMixed**, or **msoArrowheadTriangle**. Read/write **Long**.

BeginArrowheadStyle Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

BeginArrowheadWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBeginArrowheadWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBeginArrowheadWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBeginArrowheadWidthA"}

Returns or sets the width of the arrowhead at the beginning of the specified line. Can be one of the following **MsoArrowheadWidth** constants: **msoArrowheadNarrow**, **msoArrowheadWide**, **msoArrowheadWidthMedium**, or **msoArrowheadWidthMixed**. Read/write **Long**.

BeginArrowheadWidth Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

DashStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDashStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDashStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDashStyleA"}

Returns or sets the dash style for the specified line. Can be one of the following **MsoLineDashStyle** constants: **msoLineDash**, **msoLineDashDot**, **msoLineDashDotDot**, **msoLineDashStyleMixed**, **msoLineLongDash**, **msoLineLongDashDot**, **msoLineRoundDot**, **msoLineSolid**, or **msoLineSquareDot**. Read/write **Long**.

DashStyle Property Example

This example adds a blue dashed line to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

EndArrowheadLength Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndArrowheadLengthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndArrowheadLengthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndArrowheadLengthA"}

Returns or sets the length of the arrowhead at the end of the specified line. Can be one of the following **MsoArrowheadLength** constants: **msoArrowheadLengthMedium**, **msoArrowheadLengthMixed**, **msoArrowheadLong**, or **msoArrowheadShort**. Read/write **Long**.

EndArrowheadLength Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

EndArrowheadStyle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndArrowheadStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndArrowheadStyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndArrowheadStyleA"}

Returns or sets the style of the arrowhead at the end of the specified line. Can be one of the following **MsoArrowheadStyle** constants: **msoArrowheadDiamond**, **msoArrowheadNone**, **msoArrowheadOpen**, **msoArrowheadOval**, **msoArrowheadStealth**, **msoArrowheadStyleMixed**, or **msoArrowheadTriangle**. Read/write **Long**.

EndArrowheadStyle Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

EndArrowheadWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndArrowheadWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndArrowheadWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndArrowheadWidthA"}

Returns or sets the width of the arrowhead at the end of the specified line. Can be one of the following **MsoArrowheadWidth** constants: **msoArrowheadNarrow**, **msoArrowheadWide**, **msoArrowheadWidthMedium**, or **msoArrowheadWidthMixed**. Read/write **Long**.

EndArrowheadWidth Property Example

This example adds a line to `myDocument`. There's a short, narrow oval on the line's starting point and a long, wide triangle on its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(100, 100, 200, 300).Line
    .BeginArrowheadLength = msoArrowheadShort
    .BeginArrowheadStyle = msoArrowheadOval
    .BeginArrowheadWidth = msoArrowheadNarrow
    .EndArrowheadLength = msoArrowheadLong
    .EndArrowheadStyle = msoArrowheadTriangle
    .EndArrowheadWidth = msoArrowheadWide
End With
```

Style Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro StyleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro StyleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro StyleA"}

Returns or sets the line style. Can be one of the following **MsoLineStyle** constants: **msoLineSingle**, **msoLineStyleMixed**, **msoLineThickBetweenThin**, **msoLineThickThin**, **msoLineThinThick**, or **msoLineThinThin**. Read/write **Long**.

Style Property Example

This example adds a thick, blue compound line to `myDocument`. The compound line consists of a thick line with a thin line on either side of it.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .Style = msoLineThickBetweenThin
    .Weight = 8
    .ForeColor.RGB = RGB(0, 0, 255)
End With
```

Weight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproWeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproWeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproWeightA"}

Returns or sets the thickness of the specified line, in points. Read/write **Single**.

Weight Property Example

This example adds a green dashed line two points thick to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(0, 255, 255)
    .Weight = 2
End With
```

FileFind Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFileFindC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFileFindX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFileFindA"}

Returns a **FileFind** object that can be used to search for files using either an absolute or relative path. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Note This property is available only in Microsoft Office for the Macintosh.

FileFind Property Example

This example displays the names of all files in the My Documents folder that begin with "New."

```
With Application.FileFind
    .FileName = "New*.*"
    .LookIn = "C:\My Documents"
    .Execute
    For i = 1 To .FoundFiles.Count
        MsgBox .FoundFiles(i)
    Next
End With
```

Add Method (Tags Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTagsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTagsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTagsObjA"}

Creates a tag for the specified object. If the tag already exists, this method replaces the existing tag value.

Syntax

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

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

Name Required **String**. The new tag name. Use "name" as the string for this argument to set the value of the name tag.

Value Required **String**. The new tag value.

Remarks

The **Tags** object contains a pair of strings – the tag name and the tag value – for each tag. Use the **Add** method to create a tag, and use the **Name** and **Value** methods to return a tag's name and value components.

Add Method (Tags Object) Example

This example adds a tag named "Priority" and sets the value of the name tag for slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Tags
    .Add "Name", "New Figures"    'Sets value for name tag
    .Add "Priority", "Low"        'Adds "Priority" tag with value "Low"
End With
```

Colors Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthColorsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthColorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthColorsA"}

Returns an **RGBColor** object that represents a single color in a color scheme.

Syntax

expression.**Colors(*SchemeColor*)**

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

SchemeColor Required **Long**. The individual color in the specified color scheme. Can be one of the following **PpColorSchemeIndex** constants: **ppAccent1**, **ppAccent2**, **ppAccent3**, **ppBackground**, **ppFill**, **ppForeground**, **ppNotSchemeColor**, **ppShadow**, or **ppTitle**.

Colors Method Example

This example sets the title color for slides one and three in the active presentation.

```
Set mySlides = ActivePresentation.Slides.Range(Array(1, 3))
mySlides.ColorScheme.Colors(ppTitle).RGB = RGB(0, 255, 0)
```

ColorScheme Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproColorSchemeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproColorSchemeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproColorSchemeA"}

Returns or sets the **ColorScheme** object that represents the scheme colors for the specified slide, slide range, or slide master. Read/write.

ColorScheme Property Example

This example sets the title color to green for slides one and three in the active presentation.

```
Set mySlides = ActivePresentation.Slides.Range(Array(1, 3))
mySlides.ColorScheme.Colors(ppTitle).RGB = RGB(0, 255, 0)
```

DisplayComments Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDisplayCommentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDisplayCommentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDisplayCommentsA"}

True if comments are displayed in the specified presentation. Read/write **Long**.

DisplayComments Property Example

This example hides comments in the active presentation.

```
ActivePresentation.DisplayComments = False
```

Presentation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresentationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresentationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresentationA"}

Returns a **Presentation** object that represents the presentation in which the specified document window or slide show window was created. Read-only.

Remarks

If the slide that's currently displayed in document window one is from an embedded presentation, `Windows(1).View.Slide.Parent` returns the embedded presentation, and `Windows(1).Presentation` returns the presentation in which document window one was created.

If the slide that's currently displayed in slide show window one is from an embedded presentation, `SlideShowWindows(1).View.Slide.Parent` returns the embedded presentation, and `SlideShowWindows(1).Presentation` returns the presentation in which the slide show was started.

Presentation Property Example

This example continues the slide numbering for the presentation in window one into the slide numbering for the presentation in window two.

```
firstPresSlides = Windows(1).Presentation.Slides.Count
Windows(2).Presentation.PageSetup.FirstSlideNumber = firstPresSlides + 1
```

VBProject Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVBProjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVBProjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVBProjectA"}

Returns a **VBProject** object that represents the individual Visual Basic project for the presentation. Read-only.

VBProject Property Example

This example changes the name of the Visual Basic project for the active presentation.

```
ActivePresentation.VBProject.Name = "TestProject"
```

EndNamedShow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthEndNamedShowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthEndNamedShowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthEndNamedShowA"}

Switches from running a custom, or named, slide show to running the entire presentation of which the custom show is a subset. When the slide show advances from the current slide, the next slide displayed will be the next one in the entire presentation, not the next one in the custom slide show.

Syntax

expression.**EndNamedShow**

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

EndNamedShow Method Example

If a custom slide show is currently running in slide show window one, this example redefines the slide show to include all the slides in the presentation from which the slides in the custom show were selected.

```
SlideShowWindows(1).View.EndNamedShow
```

GotoNamedShow Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthGotoNamedShowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthGotoNamedShowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthGotoNamedShowA"}

Switches to the specified custom, or named, slide show during another slide show. When the slide show advances from the current slide, the next slide displayed will be the next one in the specified custom slide show, not the next one in current slide show.

Syntax

expression.**GotoNamedShow(*SlideShowName*)**

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

SlideShowName Required **String**. The name of the custom slide show to be switched to.

GotoNamedShow Method Example

This example redefines the slide show running in slide show window one to include only the slides in the custom slide show named "Quick Show."

```
SlideShowWindows(1).View.GotoNamedShow "Quick Show"
```

AddMediaObject Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddMediaObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddMediaObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddMediaObjectA"}

Creates a media object. Returns a **Shape** object that represents the new media object.

Syntax

expression.**AddMediaObject(*FileName, Left, Top, Width, Height*)**

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

FileName Required **String**. The file from which the media object is to be created. If the path isn't specified, the current working folder is used.

Left, ***Top*** Optional **Single**. The position (in points) of the upper-left corner of the media object's bounding box relative to the upper-left corner of the document.

Width, ***Height*** Optional **Single**. The width and height of the media object's bounding box, in points.

AddMediaObject Method Example

This example adds the movie named "Clock.avi" to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddMediaObject FileName:="C:\WINNT\clock.avi", Left:=5,
Top:=5, Width:=100, Height:=100
```

AddPlaceholder Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPlaceholderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPlaceholderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPlaceholderA"}

Restores a previously deleted placeholder on a slide. Returns a **Shape** object that represents the restored placeholder.

Note If you haven't previously deleted the specified placeholder, this method causes an error.

Syntax

expression.**AddPlaceholder(*Type, Left, Top, Width, Height*)**

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

Type Required **Long**. The type of placeholder to be restored. Can be one of the following **PpPlaceholderType** constants:

ppPlaceholderBitmap	**ppPlaceholderObject**
ppPlaceholderBody	**ppPlaceholderOrgChart**
ppPlaceholderCenterTitle	**ppPlaceholderSlideNumber**
ppPlaceholderChart	**ppPlaceholderSubtitle**
ppPlaceholderDate	**ppPlaceholderTable**
ppPlaceholderFooter	**ppPlaceholderTitle**
ppPlaceholderHeader	**ppPlaceholderVerticalBody**
ppPlaceholderMediaClip	**ppPlaceholderVerticalTitle**
ppPlaceholderMixed	

Placeholders of type **ppPlaceholderVerticalBody** or **ppPlaceholderVerticalTitle** are found only on slides of layout type **ppLayoutVerticalText**, **ppLayoutClipArtAndVerticalText**, **ppLayoutVerticalTitleAndText**, or **ppLayoutVerticalTitleAndTextOverChart**. You cannot create slides with any of these these layouts from the user interface; you must create them programmatically by using the **Add** method or by setting the **Layout** property of an existing slide.

Left, ***Top*** Optional **Single**. The position (in points) of the upper-left corner of the placeholder relative to the upper-left corner of the document.

Width, ***Height*** Optional **Single**. The width and height of the placeholder, in points.

Remarks

If more than one placeholder of a specified type has been deleted from the slide, the **AddPlaceholder** method will add them back to the slide, one by one, starting with the placeholder that has the lowest original index number.

AddPlaceholder Method Example

Suppose that slide two in the active presentation originally had a title at the top of the slide that's been deleted, either manually or with the following line of code.

```
ActivePresentation.Slides(2).Shapes.Placeholders(1).Delete
```

This example restores the deleted placeholder to slide two.

```
Application.ActivePresentation.Slides(2).Shapes.AddPlaceholder
ppPlaceholderTitle
```

FarEastLineBreakControl Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFarEastLineBreakControlC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFarEastLineBreakControlX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFarEastLineBreakControlA"}

Not used in the U.S. English version of PowerPoint.

HangingPunctuation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHangingPunctuationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHangingPunctuationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHangingPunctuationA"}

Not used in the U.S. English version of PowerPoint.

HasTextFrame Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHasTextFrameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHasTextFrameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHasTextFrameA"}

True if the specified shape has a text frame and can therefore contain text. Read-only **Long**.

HasTextFrame Property Example

This example extracts text from all shapes on `myDocument` that contain text frames, and then it stores the names of these shapes and the text they contain in an array.

```
Dim shpTextArray() As Variant
Dim numShapes, numAutoShapes, i As Long

Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    numShapes = .Count
    If numShapes > 1 Then
    numTextShapes = 0
        ReDim autoShpArray(1 To 2, 1 To numShapes)
        For i = 1 To numShapes
            If .Item(i).HasTextFrame Then
                numTextShapes = numTextShapes + 1
                shpTextArray(numTextShapes, 1) = .Item(i).Name
                shpTextArray(numTextShapes, 2)
= .Item(i).TextFrame.TextRange.Text
            End If
        Next
        ReDim Preserve shpTextArray(1 To 2, 1 To numAutoShapes)
    End If
End With
```

NameAscii Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNameAsciiC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNameAsciiX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNameAsciiA"}

Not used in the U.S. English version of PowerPoint.

NameFarEast Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNameFarEastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNameFarEastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNameFarEastA"}

Not used in the U.S. English version of PowerPoint.

PlaceholderFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPlaceholderFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPlaceholderFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPlaceholderFormatA"}

Returns a **PlaceholderFormat** object that contains the properties that are unique to placeholders. Read-only.

PlaceholderFormat Property Example

This example adds text to placeholder one on slide one in the active presentation if that placeholder is a horizontal title placeholder.

```
With ActivePresentation.Slides(1).Shapes.Placeholders
    If .Count > 0 Then
        With .Item(1)
            Select Case .PlaceholderFormat.Type
                Case ppPlaceholderTitle
                    .TextFrame.TextRange = "Title Text"
                Case ppPlaceholderCenterTitle
                    .TextFrame.TextRange = "Centered Title Text"
                Case Else
                    MsgBox "There's no horizontal title on this slide"
            End Select
        End With
    End If
End With
```

PrintFontsAsGraphics Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPrintFontsAsGraphicsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPrintFontsAsGraphicsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPrintFontsAsGraphicsA"}

True if TrueType fonts are printed as graphics. Read/write **Long**.

PrintFontsAsGraphics Property Example

This example specifies that TrueType fonts in the active presentation be printed as graphics.

```
ActivePresentation.PrintOptions.PrintFontsAsGraphics = True
```

SetShapesDefaultProperties Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetShapesDefaultPropertiesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetShapesDefaultPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetShapesDefaultPropertiesA"}

Applies the formatting for the specified shape to the default shape. Shapes created after this method has been used will have this formatting applied to them by default.

Syntax

expression.**SetShapesDefaultProperties**

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

SetShapesDefaultProperties Method 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 = ActivePresentation.Slides(1)
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
        .SetShapesDefaultProperties     ' Sets formatting for default
shapes
    End With
    .AddShape msoShapeRectangle, 90, 90, 40, 30  ' New shape has default
formatting
End With
```

SlideIDs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro SlideIDsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro SlideIDsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro SlideIDsA"}

Returns an array of slide IDs for the specified named slide show. Read-only **Variant**.

SlideIDs Property Example

This example adds the current slide in the active window to the custom slide show named "Marketing Short Version." Note that to save a modified version of the custom slide show, you must delete the original custom show and then add it again, using the same name. Also note that if you want to resize an array contained in a **Variant** variable, you must explicitly declare the variable before attempting to resize its array.

```
Dim customShowSlideIDs As Variant         ' NOTE - this line is NOT
optional - can't redim array without this
Dim customShowToExpand As NamedSlideShow

customShowName = "Marketing Short Version"
Set customShowToExpand =
ActivePresentation.SlideShowSettings.NamedSlideShows(customShowName)
slideToAddID = ActiveWindow.View.Slide.SlideID
customShowSlideIDs = customShowToExpand.SlideIDs
numSlides = UBound(customShowSlideIDs)

ReDim Preserve customShowSlideIDs(numSlides + 1)

customShowSlideIDs(numSlides + 1) = slideToAddID
customShowToExpand.Delete
ActivePresentation.SlideShowSettings.NamedSlideShows.Add customShowName,
customShowSlideIDs
```

AutoRotateNumbers Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoRotateNumbersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoRotateNumbersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoRotateNumbersA"}

Not used in the U.S. English version of PowerPoint.

FarEastLineBreakLevel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFarEastLineBreakLevelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFarEastLineBreakLevelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFarEastLineBreakLevelA"}

Not used in the U.S. English version of PowerPoint.

NoLineBreakAfter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro NoLineBreakAfterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro NoLineBreakAfterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro NoLineBreakAfterA"}

Not used in the U.S. English version of PowerPoint.

NoLineBreakBefore Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNoLineBreakBeforeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNoLineBreakBeforeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNoLineBreakBeforeA"}

Not used in the U.S. English version of PowerPoint.

UpdateLinks Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUpdateLinksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUpdateLinksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUpdateLinksA"}

Updates linked OLE objects in the specified presentation.

Syntax

expression.**UpdateLinks**

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

UpdateLinks Method Example

This example updates all OLE links in the active presentation.

```
ActivePresentation.UpdateLinks
```

PrintColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPrintColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPrintColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPrintColorTypeA"}

Returns or sets the way the specified document will be printed: in black and white, in pure black and white (also referred to as high contrast), or in color. Can be one of the following **PpPrintColorType** constants: **ppPrintBlackAndWhite**, **ppPrintColor**, or **ppPrintPureBlackAndWhite**. The default value is set by the printer. Read/write **Long**.

PrintColorType Property Example

This example prints the slides in the active presentation in color.

```
With Application.ActivePresentation
    .PrintOptions.PrintColorType = ppPrintColor
    .PrintOut
End With
```

Add Method (NamedSlideShows Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddNamedSlideShowsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddNamedSlideShowsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddNamedSlideShowsObjA"}

Creates a new named slide show and adds it to the collection of named slide shows in the specified presentation. Returns a **NamedSlideShow** object that represents the new named slide show.

Syntax

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

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

Name Required **String**. The name of the slide show.

SafeArrayOfSlideIDs Required **Variant**. Contains the unique slide IDs of the slides to be displayed in a slide show.

Remarks

The name you specify when you add a named slide show is the name you use as an argument to the **Run** method to run the named slide show.

Add Method (NamedSlideShows Collection Object) Example

This example adds to the active presentation a named slide show "Quick Show" that contains slides 2, 7, and 9. The example then runs this slide show.

```
Dim qSlides(1 To 3) As Long
With ActivePresentation
    With .Slides
        qSlides(1) = .Item(2).SlideID
        qSlides(2) = .Item(7).SlideID
        qSlides(3) = .Item(9).SlideID
    End With
    With .SlideShowSettings
        .NamedSlideShows.Add "Quick Show", qSlides
        .RangeType = ppShowNamedSlideShow
        .SlideShowName = "Quick Show"
        .Run
    End With
End With
```

Application Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjApplicationC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjApplicationP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjApplicationM"}

Application

└

Multiple Objects

Represents the entire Microsoft PowerPoint application. The **Application** object contains:

- Application-wide settings and options (the name of the active printer, for example).
- Properties that return top-level objects, such as **ActivePresentation**, **Windows**, and so on.

Using the Application Object

Use the **Application** property to return the **Application** object. The following example applies the **Windows** property to the **Application** object.

```
Application.Windows(2).Activate
```

The following example creates a PowerPoint **Application** object in another application, starts PowerPoint (if it's not already running), and opens an existing presentation named "Ex_a2a.ppt."

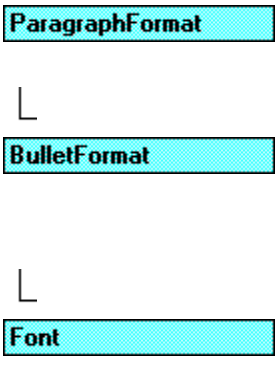
```
Set ppt = CreateObject("Powerpoint.Application.8")
ppt.Visible = True
ppt.Presentations.Open "c:\My Documents\ex_a2a.ppt"
```

Remarks

When you are writing code that will run from PowerPoint, the following properties of the **Application** object can be used without the object qualifier: **ActivePresentation**, **ActiveWindow**, **AddIns**, **Assistant**, **CommandBars**, **Presentations**, **SlideShowWindows**, **Windows**. For example, instead of writing `Application.ActiveWindow.Height = 200`, you can write `ActiveWindow.Height = 200`.

BulletFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjBulletFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjBulletFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjBulletFormatM"}



Represents bullet formatting.

Using the BulletFormat Object

Use the **Bullet** property to return the **BulletFormat** object. The following example sets the bullet size and color for the paragraphs in shape two on slide one in the active presentation.

```
With ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat.Bullet
        .Visible = True
        .RelativeSize = 1.25
        .Character = 169
        With .Font
            .Color.RGB = RGB(255, 255, 0)
            .Name = "Symbol"
        End With
    End With
End With
```

ColorSchemes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjColorSchemesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjColorSchemesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjColorSchemesM"}

A collection of all the **ColorScheme** objects in the specified presentation. Each **ColorScheme** object represents a color scheme, which is a set of colors that are used together on a slide.

Using the ColorSchemes Collection

Use the **ColorSchemes** property to return the **ColorSchemes** collection. Use **ColorSchemes**(*index*), where index is the color scheme index number, to return a single **ColorScheme** object. The following example deletes color scheme two from the active presentation.

```
ActivePresentation.ColorSchemes(2).Delete
```

Use the **Add** method to create a new color scheme and add it to the **ColorSchemes** collection. The following example adds a color scheme to the active presentation and sets the title color and background color for the color scheme (because no argument was used with the **Add** method, the added color scheme is initially identical to the first standard color scheme in the presentation).

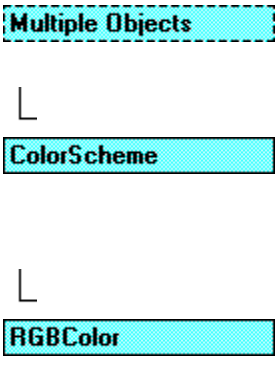
```
With ActivePresentation.ColorSchemes.Add
    .Colors(ppTitle).RGB = RGB(255, 0, 0)
    .Colors(ppBackground).RGB = RGB(128, 128, 0)
End With
```

Set the **ColorScheme** property of a **Slide**, **SlideRange**, or **Master** object to return the color scheme for one slide, a set of slides, or a master, respectively. The following example sets the color scheme for all the slides in the active presentation to the third color scheme in the presentation.

```
With ActivePresentation
    .Slides.Range.ColorScheme = .ColorSchemes(3)
End With
```

ColorScheme Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjColorSchemeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjColorSchemeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjColorSchemeM"}



Represents a color scheme, which is a set of eight colors used for the different elements of a slide, notes page, or handout, such as the title or background. (Note that the color schemes for slides, notes pages, and handouts in a presentation can be set independently.) Each color is represented by an **RGBColor** object. The **ColorScheme** object is a member of the **ColorSchemes** collection. The **ColorSchemes** collection contains all the color schemes in a presentation.

Using the ColorScheme Object

This section describes how to do the following:

- Return a **ColorScheme** object from the collection of all the color schemes in the presentation
- Return the **ColorScheme** object attached to a specific slide or master
- Return the color of a single slide element from a **ColorScheme** object

Returning a ColorScheme object from the collection of all the color schemes in the presentation

Use **ColorSchemes**(*index*), where *index* is the color scheme index number, to return a single **ColorScheme** object. The following example deletes color scheme two from the active presentation.

```
ActivePresentation.ColorSchemes(2).Delete
```

Returning the ColorScheme object attached to a specific slide or master

Set the **ColorScheme** property of a **Slide**, **SlideRange**, or **Master** object to return the color scheme for one slide, a set of slides, or a master, respectively. The following example creates a color scheme based on the current slide, adds the new color scheme to the collection of standard color schemes for the presentation, and sets the color scheme for the slide master to the new color scheme. All new slides based on the master will have this color scheme.

```
Set newScheme = ActiveWindow.View.Slide.ColorScheme
newScheme.Colors(ppTitle).RGB = RGB(0, 150, 250)
Set newStandardScheme = ActivePresentation.ColorSchemes.Add(newScheme)
ActivePresentation.SlideMaster.ColorScheme = newStandardScheme
```

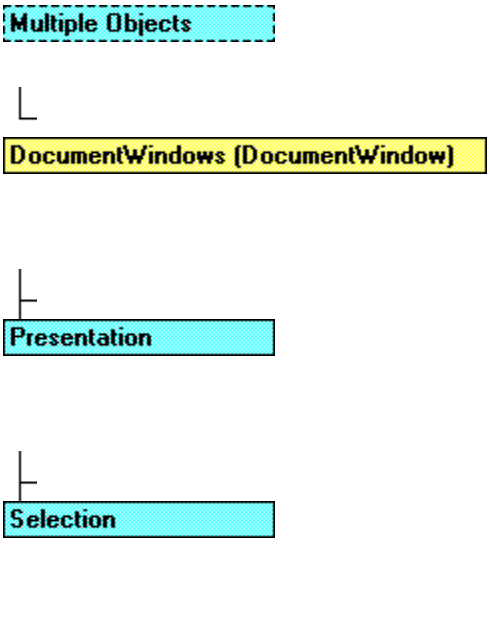
Returning the color of a single slide element from a ColorScheme object

Use the **Colors** method to return an **RGBColor** object that represents the color of a single slide-element type. You can set an **RGBColor** object to another **RGBColor** object, or you can use the **RGB** property to set or return the explicit red-green-blue (RGB) value for an **RGBColor** object. The following example sets the background color in color scheme one to red and sets the title color to the title color that's defined for color scheme two.

```
With ActivePresentation.ColorSchemes
    .Item(1).Colors(ppBackground).RGB = RGB(255, 0, 0)
    .Item(1).Colors(ppTitle) = .Item(2).Colors(ppTitle)
End With
```

DocumentWindow Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjDocumentWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjDocumentWindowP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjDocumentWindowM"}



Represents a document window. The **DocumentWindow** object is a member of the **DocumentWindows** collection. The **DocumentWindows** collection contains all the open document windows.

Using the DocumentWindow Object

Use **Windows**(*index*), where *index* is the document window index number, to return a single **DocumentWindow** object. The following example activates document window two.

```
Windows(2).Activate
```

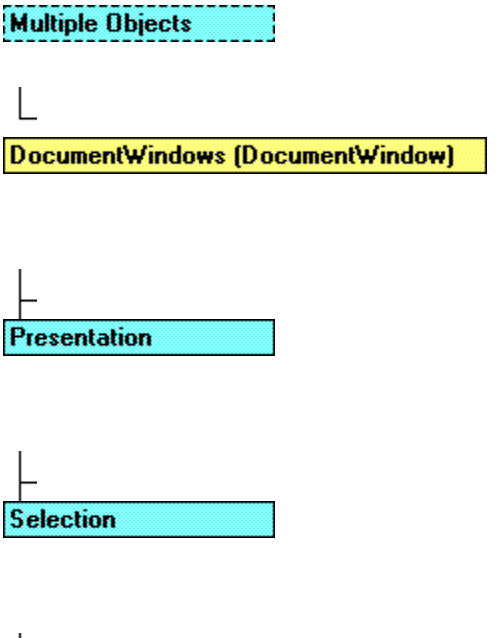
Use the **ActiveWindow** property to return the active document window. The following example maximizes the active window.

```
ActiveWindow.WindowState = ppWindowMaximized
```

Use the **Presentation** property to return the presentation that's currently running in the specified document window. Use the **Selection** property to return the selection. Use the **View** property to return the view in the specified document window.

DocumentWindows Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjDocumentWindowsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjDocumentWindowsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjDocumentWindowsM"}



A collection of all the **DocumentWindow** objects that are currently open in PowerPoint. This collection doesn't include open slide show windows, which are included in the **SlideShowWindows** collection.

Using the DocumentWindows Collection

Use the **Windows** property to return the **DocumentWindows** collection. The following example tiles the open document windows.

```
Windows.Arrange ppArrangeTiled
```

Use the **NewWindow** method to create a document window and add it to the **DocumentWindows** collection. The following example creates a new window for the active presentation.

```
ActivePresentation.NewWindow
```

Use **Windows**(*index*), where *index* is the window index number, to return a single **DocumentWindow** object. The following example closes document window two.

```
Windows(2).Close
```

ExtraColors Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjExtraColorsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjExtraColorsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjExtraColorsM"}

Presentation

└

ExtraColors

Represents the extra colors in a presentation. The object can contain up to eight colors, each of which is represented by an red-green-blue (RGB) value.

Using the ExtraColors Object

Use the **ExtraColors** property to return the **ExtraColors** object. Use **ExtraColors**(*index*), where index is the extra color index number, to return the red-green-blue (RGB) value for a single extra color. The following example adds a rectangle to slide one in the active presentation and sets its fill foreground color to the first extra color. If there hasn't been at least one extra color defined for the presentation, this example will fail.

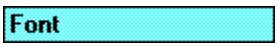
```
With ActivePresentation
    Set rect = .Slides(1).Shapes.AddShape(msoShapeRectangle, 50, 50, 100,
200)
    rect.Fill.ForeColor.RGB = .ExtraColors(1)
End With
```

Use the **Add** method to add an extra color. The following example adds an extra color to the active presentation (if the color hasn't already been added).

```
ActivePresentation.ExtraColors.Add RGB(69, 32, 155)
```

Font Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjFontC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjFontP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjFontM"}



Represents character formatting for text or a bullet. The **Font** object is a member of the **Fonts** collection. The **Fonts** collection contains all the fonts used in a presentation.

Using the Font Object

This section describes how to do the following:

- Return the **Font** object that represents the font attributes of a specified bullet, a specified range of text, or all text at a specified outline level
- Return a **Font** object from the collection of all the fonts used in the presentation

Returning the Font object that represents the font attributes of a specified bullet, a specified range of text, or all text at a specified outline level

Use the **Font** property to return the **Font** object that represents the font attributes for a specific bullet, text range, or outline level. The following example sets the title text on slide one and sets the font properties.

```
With ActivePresentation.Slides(1).Shapes.Title.TextFrame.TextRange
    .Text = "Volcano Coffee"
    With .Font
        .Italic = True
        .Name = "Palatino"
        .Color.RGB = RGB(0, 0, 255)
    End With
End With
```

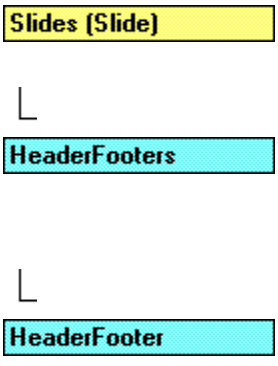
Returning a Font object from the collection of all the fonts used in the presentation

Use **Fonts**(*index*), where *index* is the font's name or index number, to return a single **Font** object. The following example checks to see whether font one in the active presentation is embedded in the presentation.

```
If ActivePresentation.Fonts(1).Embedded = True Then MsgBox "Font 1 is
embedded"
```

HeaderFooter Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjHeaderFooterC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjHeaderFooterP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjHeaderFooterM"}



Represents a header, footer, date and time, slide number, or page number on a slide or master. All the **HeaderFooter** objects for a slide or master are contained in a **HeadersFooters** object.

Using the HeaderFooter Object

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

Use this property	To return
DateAndTime	A **HeaderFooter** object that represents the date and time on the slide.
Footer	A **HeaderFooter** object that represents the footer for the slide.
Header	A **HeaderFooter** object that represents the header for the slide. This works only for notes pages and handouts, not for slides.
SlideNumber	A **HeaderFooter** object that represent the slide number (on a slide) or page number (on a notes page or a handout).

Note **HeaderFooter** objects aren't available for **Slide** objects that represent notes pages. The **HeaderFooter** object that represents a header is available only for a notes master or handout master.

You can set properties of **HeaderFooter** objects for single slides. The following example sets the footer text for slide one in the active presentation.

```
ActivePresentation.Slides(1).HeadersFooters.Footer.Text = "Volcano Coffee"
```

You can also set properties of **HeaderFooter** objects for the slide master, title master, notes master, or handout master to affect all slides, title slides, notes pages, or handouts and outlines at the same time. The following example sets the text for the footer in the slide master for the active presentation, sets the format for the date and time, and turns on the display of slide numbers. These settings will apply to all slides that are based on this master that display master graphics and that have not had their footer and date and time set individually.

```
Set mySlidesHF = ActivePresentation.SlideMaster.HeadersFooters
With mySlidesHF
    .Footer.Visible = True
    .Footer.Text = "Regional Sales"
    .SlideNumber.Visible = True
    .DateAndTime.Visible = True
    .DateAndTime.UseFormat = True
```

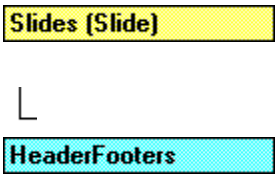
```
    .DateAndTime.Format = ppDateTimeMdyy
End With
```

To clear header and footer information that has been set for individual slides and make sure all slides display the header and information you define for the slide master, run the following code before running the previous example.

```
For Each s In ActivePresentation.Slides
    s.DisplayMasterShapes = True
    s.HeadersFooters.Clear
Next
```

HeadersFooters Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjHeadersFootersC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjHeadersFootersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjHeadersFootersM"}



Contains all the **HeaderFooter** objects on the specified slide, notes page, handout, or master. Each **HeaderFooter** object represents a header, footer, date and time, or slide number.

Note **HeaderFooter** objects aren't available for **Slide** objects that represent notes pages. The **HeaderFooter** object that represents a header is available only for a notes master or handout master.

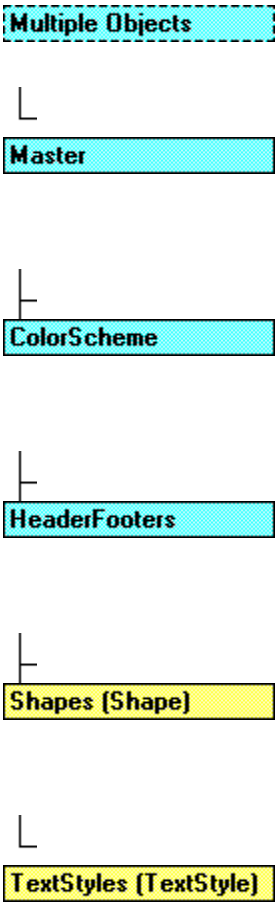
Using the HeaderFooters Object

Use the **HeadersFooters** property to return the **HeadersFooters** object. Use the **DateAndTime**, **Footer**, **Header**, or **SlideNumber** property to return an individual **HeaderFooter** object. The following example sets the footer text for slide one in the active presentation.

```
ActivePresentation.Slides(1).HeadersFooters.Footer.Text = "Volcano Coffee"
```

Master Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjMasterP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjMasterM"}



Represents a slide master, title master, handout master, or notes master.

Using the Master Object

To return a **Master** object, use the **Master** property of the **Slide** object or **SlideRange** collection, or use the **HandoutMaster**, **NotesMaster**, **SlideMaster**, or **TitleMaster** property of the **Presentation** object. The following example sets the background fill for the slide master for the active presentation.

```
ActivePresentation.SlideMaster.Background.Fill.PresetGradient
msoGradientHorizontal, 1, msoGradientBrass
```

To add a title master to a presentation and return a **Master** object that represents the new title master, use the **AddTitleMaster** method. The following example adds a title master to the active presentation and places the title placeholder 10 points from the top of the master.

```
ActivePresentation.AddTitleMaster.Shapes.Title.Top = 10
```

NamedSlideShow Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjNamedSlideShowC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjNamedSlideShowP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjNamedSlideShowM"}

SlideShowSettings

└

NamedSlideShows (NamedSlideShow)

Represents a custom slide show, which is a named subset of slides in a presentation. The **NamedSlideShow** object is a member of the **NamedSlideShows** collection. The **NamedSlideShows** collection contains all the named slide shows in the presentation.

Using the NamedSlideShow Object

Use **NamedSlideShows**(*index*), where *index* is the custom slide show name or index number, to return a single **NamedSlideShow** object. The following example deletes the custom slide show named "Quick Show."

```
ActivePresentation.SlideShowSettings.NamedSlideShows("Quick Show").Delete
```

Use the **SlideIDs** property to return an array that contains the unique slide IDs for all the slides in the specified custom show. The following example displays the slide IDs for the slides in the custom slide show named "Quick Show."

```
idArray = ActivePresentation.SlideShowSettings.NamedSlideShows("Quick
Show").SlideIDs
For i = 1 To UBound(idArray)
    MsgBox idArray(i)
Next
```

NamedSlideShows Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjNamedSlideShowsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjNamedSlideShowsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjNamedSlideShowsM"}

SlideShowSettings

└

NamedSlideShows (NamedSlideShow)

A collection of all the **NamedSlideShow** objects in the presentation. Each **NamedSlideShow** object represents a custom slide show.

Using the NamedSlideShows Collection

Use the **NamedSlideShows** property to return the **NamedSlideShows** collection. Use **NamedSlideShows**(*index*), where *index* is the custom slide show name or index number, to return a single **NamedSlideShow** object. The following example deletes the custom slide show named "Quick Show."

```
ActivePresentation.SlideShowSettings.NamedSlideShows("Quick Show").Delete
```

Use the **Add** method to create a new slide show and add it to the **NamedSlideShows** collection.The following example adds to the active presentation the named slide show "Quick Show" that contains slides 2, 7, and 9. The example then runs this custom slide show.

```
Dim qSlides(1 To 3) As Long
With ActivePresentation
    With .Slides
        qSlides(1) = .Item(2).SlideID
        qSlides(2) = .Item(7).SlideID
        qSlides(3) = .Item(9).SlideID
    End With
    With .SlideShowSettings
        .NamedSlideShows.Add "Quick Show", qSlides
        .RangeType = ppShowNamedSlideShow
        .SlideShowName = "Quick Show"
        .Run
    End With
End With
```

PageSetup Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPageSetupC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPageSetupP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPageSetupM"}

Presentation

└

PageSetup

Contains information about the page setup for slides, notes pages, handouts, and outlines in a presentation.

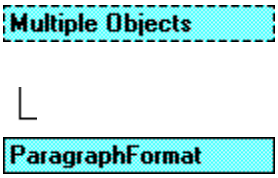
Using the PageSetup Object

Use the **PageSetup** property to return the **PageSetup** object. The following example sets all slides in the active presentation to be 11 inches wide and 8.5 inches high and sets the slide numbering for the presentation to start at 17.

```
With ActivePresentation.PageSetup
    .SlideWidth = 11 * 72
    .SlideHeight = 8.5 * 72
    .FirstSlideNumber = 17
End With
```

ParagraphFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjParagraphFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjParagraphFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjParagraphFormatM"}



Represents the paragraph formatting of a text range.

Using the ParagraphFormat Object

Use the **ParagraphFormat** property to return the **ParagraphFormat** object. The followng example left aligns the paragraphs in shape two on slide one in the active presentation.

```
ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .ParagraphFormat.Alignment = ppAlignLeft
```

Placeholders Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPlaceholdersC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPlaceholdersP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPlaceholdersM"}

Shapes (Shape)

└

Placeholders (Shape)

A collection of all the **Shape** objects that represent placeholders on the specified slide. Each **Shape** object in the **Placeholders** collection represents a placeholder for text, a chart, a table, an organizational chart, or some other type of object. If the slide has a title, the title is the first placeholder in the collection.

Using the Placeholders Collection

Use the **Placeholders** property to return the **Placeholders** collection. Use **Placeholders**(*index*), where *index* is the placeholder index number, to return a **Shape** object that represents a single placeholder. Note that for any slide that has a title, `Shapes.Title` is equivalent to `Shapes.Placeholders(1)`.The following example adds a new slide with a Bulleted List slide layout to the beginning of the presentation, sets the text for the title, and then adds two paragraphs to the text placeholder.

```
Set sObj = ActivePresentation.Slides.Add(1, ppLayoutText).Shapes
sObj.Title.TextFrame.TextRange.Text = "This is the title text"
sObj.Placeholders(2).TextFrame.TextRange.Text = "Item 1" & Chr(13) & "Item
2"
```

You can delete individual placeholders by using the **Delete** method, and you can restore deleted placeholders by using the **AddPlaceholder** method, but you cannot add any more placeholders to a slide than it had when it was created. To change the number of placeholders on a given slide, set the **Layout** property.

PlaceholderFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPlaceholderFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPlaceholderFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPlaceholderFormatM"}

Shapes (Shape)

L

PlaceholderFormat

Contains properties that apply specifically to placeholders, such as placeholder type.

Using the PlaceholderFormat Object

Use the **PlaceholderFormat** property to return a **PlaceholderFormat** object. The following example adds text to placeholder one on slide one in the active presentation if that placeholder exists and is a horizontal title placeholder.

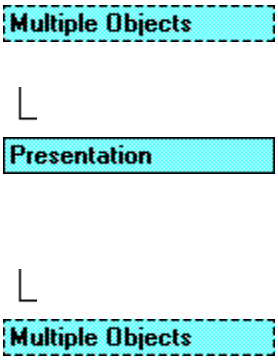
```
With ActivePresentation.Slides(1).Shapes.Placeholders
    If .Count > 0 Then
        With .Item(1)
            Select Case .PlaceholderFormat.Type
                Case ppPlaceholderTitle
                    .TextFrame.TextRange = "Title Text"
                Case ppPlaceholderCenterTitle
                    .TextFrame.TextRange = "Centered Title Text"
                Case Else
                    MsgBox "There's no horizontal title on this slide"
            End Select
        End With
    End If
End With
```

placeholders

The boxes with dotted outlines that appear on newly created slides. These boxes serve as placeholders for objects such as slide titles, text, charts, tables, organizational charts, and clip art.

Presentation Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPresentationC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPresentationP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPresentationM"}



Represents a PowerPoint presentation. The **Presentation** object is a member of the **Presentations** collection. The **Presentations** collection contains all the **Presentation** objects that represent open presentations in PowerPoint.

Using the Presentation Object

This section describes how to:

- Return a presentation that you specify by name or index number
- Return the presentation in the active window
- Return the presentation in any document window or slide show window you specify

Returning a presentation that you specify by name or index number

Use **Presentations**(*index*), where *index* is the presentation's name or index number, to return a single **Presentation** object. The name of the presentation is the file name, with or without the file name extension, and without the path. The following example adds a slide to the beginning of Sample Presentation.

```
Presentations("Sample Presentation").Slides.Add 1, 1
```

Note that if multiple presentations with the same name are open, the first presentation in the collection with the specified name is returned.

Returning the presentation in the active window

Use the **ActivePresentation** property to return the presentation in the active window. The following example saves the active presentation.

```
ActivePresentation.Save
```

Note that if an embedded presentation is in-place active, the **ActivePresentation** property returns the embedded presentation.

Returning the presentation in any document window or slide show window you specify

Use the **Presentation** property to return the presentation that's in the specified document window or slide show window. The following example displays the name of the slide show running in slide show window one.

```
MsgBox SlideShowWindows(1).Presentation.Name
```

Presentations Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPresentationsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPresentationsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPresentationsM"}

Application

└

Presentations (Presentation)

└

Multiple Objects

A collection of all the **Presentation** objects in PowerPoint. Each **Presentation** object represents a presentation that's currently open in PowerPoint.

Using the Presentations Collection

Use the **Presentations** property to return the **Presentations** collection. Use the **Add** method to create a new presentation and add it to the collection. The following example creates a new presentation, adds a slide to the presentation, and then saves the presentation.

```
Set newPres = Presentations.Add(True)
newPres.Slides.Add 1, 1
newPres.SaveAs "Sample"
```

Use **Presentations**(*index*), where *index* is the presentation's name or index number,e to return a single **Presentation** object. The following example prints presentation one.

```
Presentations(1).PrintOut
```

Use the **Open** method to open a presentation and add it to the **Presentations** collection. The following example opens the file Sales.ppt as a read-only presentation.

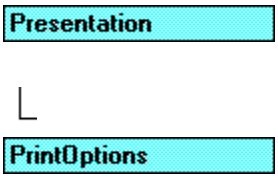
```
Presentations.Open FileName:="sales.ppt", ReadOnly:=True
```

Remarks

The **Presentations** collection doesn't include open add-ins, which are a special kind of hidden presentation. You can, however, return a single open add-in if you know its file name. For example `Presentations("oscar.ppa")` will return the open add-in named "Oscar.ppa" as a **Presentation** object. However, it is recommended that the **AddIns** collection be used to return open add-ins.

PrintOptions Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPrintOptionsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPrintOptionsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPrintOptionsM"}



Contains print options for a presentation.

Note Specifying the optional arguments ***From***, ***To***, ***Copies***, and ***Collate*** for the **PrintOut** method will set the corresponding properties of the **PrintOptions** object.

Using the PrintOptions Object

Use the **PrintOptions** property to return the **PrintOptions** object. The following example prints two uncollated color copies of all the slides (whether visible or hidden) in the active presentation. The example also scales each slide to fit the printed page and frames each slide with a thin border.

```
With ActivePresentation
    With .PrintOptions
        .NumberOfCopies = 2
        .Collate = False
        .PrintColorType = ppPrintColor
        .PrintHiddenSlides = True
        .FitToPage = True
        .FrameSlides = True
        .OutputType = ppPrintOutputSlides
    End With
    .PrintOut
End With
```

Use the **RangeType** property to specify whether to print the entire presentation or only a specified part of it. If you want to print only certain slides, set the **RangeType** property to **ppPrintSlideRange**, and use the **Ranges** property to specify which pages to print.The following example prints slides 1, 4, 5, and 6 in the active presentation

```
With ActivePresentation
    With .PrintOptions
        .RangeType = ppPrintSlideRange
        With .Ranges
            .Add 1, 1
            .Add 4, 6
        End With
    End With
    .PrintOut
End With
```

PrintRange Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPrintRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPrintRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPrintRangeM"}

PrintOptions

└

PrintRanges (PrintRange)

Represents a single range of consecutive slides or pages to be printed. The **PrintRange** object is a member of the **PrintRanges** collection. The **PrintRanges** collection contains all the print ranges that have been defined for the specified presentation.

Using the PrintRange Object

Use **Ranges**(*index*), where *index* is the print range index number, to return a single **PrintRange** object. The following example displays a message that indicates the starting and ending slide numbers for print range one in the active presentation.

```
With ActivePresentation.PrintOptions.Ranges
    If .Count > 0 Then
        With .Item(1)
            MsgBox "Print range 1 starts on slide " & .Start & _
                " and ends on slide " & .End
        End With
    End If
End With
```

Use the **Add** method to create a **PrintRange** object and add it to the **PrintRanges** collection. The following example defines three print ranges that represent slide 1, slides 3 through 5, and slides 8 and 9 in the active presentation and then prints the slides in these ranges.

```
With ActivePresentation.PrintOptions
    .RangeType = ppPrintSlideRange
    With .Ranges
        .ClearAll
        .Add 1, 1
        .Add 3, 5
        .Add 8, 9
    End With
End With
ActivePresentation.PrintOut
```

Remarks

You can set print ranges in the **PrintRanges** collection independent of the **RangeType** setting; these ranges are retained as long as the presentation they're contained in is loaded. The ranges in the **PrintRanges** collection are applied when the **RangeType** property is set to **ppPrintSlideRange**.

PrintRanges Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPrintRangesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPrintRangesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPrintRangesM"}

PrintOptions

└

PrintRanges (PrintRange)

A collection of all the **PrintRange** objects in the specified presentation. Each **PrintRange** object represents a range of consecutive slides or pages to be printed.

Using the PrintRanges Collection

Use the **Ranges** property to return the **PrintRanges** collection. The following example clears all previously defined print ranges from the collection for the active presentation.

```
ActivePresentation.PrintOptions.Ranges.ClearAll
```

Use the **Add** method to create a **PrintRange** object and add it to the **PrintRanges** collection. The following example defines three print ranges that represent slide 1, slides 3 through 5, and slides 8 and 9 in the active presentation and then prints the slides in these ranges.

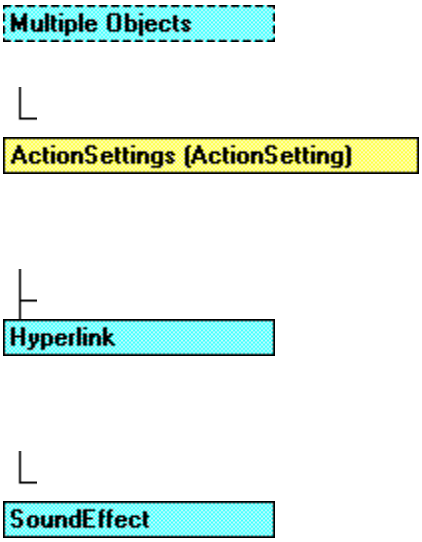
```
With ActivePresentation.PrintOptions
    .RangeType = ppPrintSlideRange
    With .Ranges
        .ClearAll
        .Add 1, 1
        .Add 3, 5
        .Add 8, 9
    End With
End With
ActivePresentation.PrintOut
```

Use **Ranges**(*index*), where *index* is the print range index number, to return a single **PrintRange** object. The following example displays a message that indicates the starting and ending slide numbers for print range one in the active presentation.

```
With ActivePresentation.PrintOptions.Ranges
    If .Count > 0 Then
        With .Item(1)
            MsgBox "Print range 1 starts on slide " & .Start & _
                " and ends on slide " & .End
        End With
    End If
End With
```

ActionSetting Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjActionSettingC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjActionSettingP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjActionSettingM"}



Contains information about how the specified shape or text range reacts to mouse actions during a slide show. The **ActionSetting** object is a member of the **ActionSettings** collection. The **ActionSettings** collection contains one **ActionSetting** object that represents how the specified object reacts when the user clicks it during a slide show and one **ActionSetting** object that represents how the specified object reacts when the user moves the mouse pointer over it during a slide show.

Using the ActionSetting Object

Use **ActionSettings**(*index*), where *index* is the either **ppMouseClick** or **ppMouseOver**, to return a single **ActionSetting** object. The following example sets the mouse-click action for the text in the third shape on slide one in the active presentation to an Internet link.

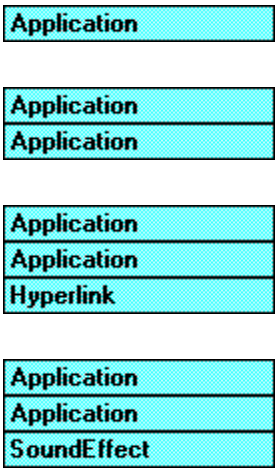
```
With
ActivePresentation.Slides(1).Shapes(3).TextFrame.TextRange.ActionSettings(p
pMouseClick)
    .Action = ppActionHyperLink
    .HyperLink.Address = "http://www.microsoft.com"
End With
```

Remarks

If you've set properties of the **ActionSetting** object that don't seem to be taking effect, make sure that you've set the **Action** property to the appropriate value.

ActionSettings Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjActionSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjActionSettingsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjActionSettingsM"}



A collection that contains the two **ActionSetting** objects for a shape or text range. One **ActionSetting** object represents how the specified object reacts when the user clicks it during a slide show, and the other **ActionSetting** object represents how the specified object reacts when the user moves the mouse pointer over it during a slide show.

Using the ActionSettings Collection

Use the **ActionSettings** property to return the **ActionSettings** collection. Use **ActionSettings**(*index*), where *index* is either **ppMouseClick** or **ppMouseOver**, to return a single **ActionSetting** object. The following example specifies that the CalculateTotal macro be run whenever the mouse pointer passes over the shape during a slide show.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseOver)
    .Action = ppActionRunMacro
    .Run = "CalculateTotal"
    .AnimateAction = True
End With
```

AddIn Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjAddInC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppobjAddInX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjAddInP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjAddInM"}

Represents a single add-in, either loaded or not loaded. The **AddIn** object is a member of the **AddIns**collection. The **AddIns** collection contains a list of all the add-ins available to Microsoft PowerPoint, regardless of whether or not they're loaded. This list corresponds to the list of add-ins displayed in the **Add-Ins** dialog box (**Tools** menu).

Using the AddIn Object

Use **AddIns**(*index*), where *index* is the add-in's title or index number, to return a single **AddIn** object. The following example loads the My Ppt Tools add-in.

```
AddIns("my ppt tools").Loaded = True
```

Don't confuse the add-in title, which appears in the **Add-Ins** dialog box, with the add-in name, which is the file name of the add-in. You must spell the add-in title exactly as it's spelled in the **Add-Ins** dialog box, but the capitalization doesn't have to match.

The index number represents the position of the add-in in the **Add-ins available** box in the **Add-Ins** dialog box. The following example displays the names of all the add-ins that are currently loaded in PowerPoint.

```
For i = 1 To AddIns.Count
    If AddIns(i).Loaded Then MsgBox AddIns(i).Name
Next
```

Remarks

Use the **Add** method to add an add-in to the list of available add-ins. Note, however, that using this method doesn't load the add-in. To load the add-in, set the **Loaded** property of the add-in to **True** after you use the **Add** method. You can perform both of these actions in a single step, as shown in the following example (note that you use the name of the add-in, not its title, with the **Add** method).

```
AddIns.Add("generic.ppa").Loaded = True
```

Use **Presentations**(*index*), where *index* is the add-in's file name (not its title), to return a reference to the presentation that corresponds to a loaded add-in. You must use the file name, because loaded add-ins usually don't appear in the **Presentations** collection. The following example sets the `presAddin` variable to the presentation for Myaddin.ppa.

```
Set presAddin = Presentations("myaddin.ppa")
```

The following example sets the `pres` variable to the presentation for the My Ppt Tools add-in.

```
Set presAddin = Presentations(AddIns("my ppt tools").Name)
```

AddIns Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjAddInsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppobjAddInsX":1} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjAddInsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjAddInsM"}

A collection of **AddIn** objects that represent all the add-ins available to Microsoft PowerPoint, regardless of whether or not they're loaded. This list corresponds to the list of add-ins displayed in the **Add-Ins** dialog box (**Tools** menu).

Using the AddIns Collection

Use the **AddIns** method to return the **AddIns** collection. The following example displays the names of all the add-ins that are currently loaded in PowerPoint.

```
For Each ad In AddIns
    If ad.Loaded Then MsgBox ad.Name
Next
```

Use the **Add** method to add an add-in to the list of available add-ins. The **Add** method adds an add-in to the list but doesn't load the add-in. To load the add-in, set the **Loaded** property of the add-in to **True** after you use the **Add** method. You can perform these two actions in a single step, as shown in the following example (note that you use the name of the add-in, not its title, with the **Add** method).

```
AddIns.Add("generic.ppa").Loaded = True
```

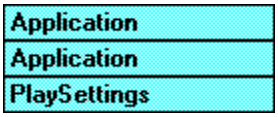
Use **AddIns**(*index*), where *index* is the add-in's title or index number, to return a single **AddIn** object. The following example loads the My Ppt Tools add-in.

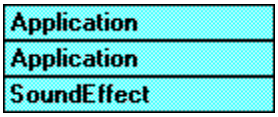
```
AddIns("my ppt tools").Loaded = True
```

Don't confuse the add-in title, which appears in the **Add-Ins** dialog box, with the add-in name, which is the file name of the add-in. You must spell the add-in title exactly as it's spelled in the **Add-Ins** dialog box, but the capitalization doesn't have to match.

AnimationSettings Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjAnimationSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjAnimationSettingsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjAnimationSettingsM"}





Represents the special effects applied to the animation for the specified shape during a slide show.

Using the AnimationSettings Object

Use the **AnimationSettings** property of the **Shape** object to return the **AnimationSettings** object. The following example adds a slide that contains both a title and a three-item list to the active presentation, and then it sets the list to be animated by first-level paragraphs, to fly in from the left when animated, to dim to the specified color after being animated, and to animate its items in reverse order.

```
Set sObjs = ActivePresentation.Slides.Add(2, ppLayoutText).Objects
sObjs.Title.Text = "Top Three Reasons"
With sObjs.Placeholders(1)
    .Text = "Reason 1" & Chr(13) & "Reason 2" & Chr(13) & "Reason 3"
    With .AnimationSettings
        .TextLevelEffect = ppAnimateByFirstLevel
        .EntryEffect = ppEffectFlyFromLeft
        .AfterEffect = ppAfterEffectDim
        .DimColor.RGB = RGB(100, 120, 100)
        .AnimateTextInReverse = True
    End With
End With
```

Fonts Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjFontsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjFontsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjFontsM"}

A collection of all the **Font** objects in the specified presentation. Each **Font** object represents a font that's used in the presentation.

Note The **Fonts** collection is used by the Geni Wizard to determine whether any of the fonts in the specified presentation won't be supported when Genigraphics images the slides. If you just want to set character formatting for a particular bullet or text range, use the **Font** property to return the **Font** object for thebullet or text range.

Using the Fonts Object

Use the **Fonts** property to return the **Fonts** collection. The following example displays the number of fonts used in the active presentation.

```
MsgBox ActivePresentation.Fonts.Count
```

Use **Fonts**(*index*), where *index* is the font's name or index number, to return a single **Font** object. The following example checks to see whether font one in the active presentation is embedded in the presentation.

```
If ActivePresentation.Fonts(1).Embedded = True Then MsgBox "Font 1 is
embedded"
```

HyperLink Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjHyperLinkC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjHyperLinkP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjHyperLinkM"}

Application

Application
Hyperlink

Represents a hyperlink associated with a shape. You can use a hyperlink to jump to an Internet or intranet site, to another file, or to a slide within the active presentation. The **HyperLink** object is a member of the **HyperLinks** collection. The **HyperLinks** collection contains all the hyperlinks on a slide or a master.

Using the HyperLink Object

Use the **Hyperlink** property to return a hyperlink for a shape. A shape can have two different hyperlinks assigned to it: one that's followed when the user clicks the shape during a slide show, and another that's followed when the user passes the mouse pointer over the shape during a slide show. For the hyperlink to be active during a slide show, the **Action** property must be set to **ppActionHyperlink**. The following example sets the mouse-click action for shape three on slide one in the active presentation to an Internet link.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseClick)
    .Action = ppActionHyperLink
    .HyperLink.Address = "http://www.microsoft.com"
End With
```

A slide can have more than one hyperlink. Use **Hyperlinks**(*index*), where *index* is the hyperlink number, to return a single **Hyperlink** object. The following example adds a shortcut to the target document for the mouse-click hyperlink for shape three to the Favorites folder.

```
ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseClick).HyperLi
nk.AddToFavorites
```

HyperLinks Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjHyperLinksC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjHyperLinksP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjHyperLinksM"}

A collection of all the **HyperLink** objects on a slide or master.

Using the HyperLinks Collection

Use the **HyperLinks** property to return the **HyperLinks** collection. The following example updates all hyperlinks on slide one in the active presentation that have the specified address.

```
For Each hl In ActivePresentation.Slides(1).Hyperlinks
    If hl.Name = "c:\current work\sales.ppt" Then hl.Name = "c:\new\
newsales.ppt"
Next
```

Use the **Hyperlink** property to create a hyperlink and add it to the **Hyperlinks** collection. The following example sets a hyperlink that will be followed when the user clicks shape three on slide one in the active presentation during a slide show and adds the new hyperlink to the collection. Note that if shape three already has a mouse-click hyperlink defined, the following example will delete this hyperlink from the collection when it adds the new one, so the number of items in the **Hyperlinks** collection won't change.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseClick)
    .Action = ppActionHyperLink
    .HyperLink.Address = "http://www.microsoft.com"
End With
```

PlaySettings Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPlaySettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPlaySettingsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPlaySettingsM"}

AnimationSettings

Application
PlaySettings

Contains information about how the specified media clip will be played during a slide show.

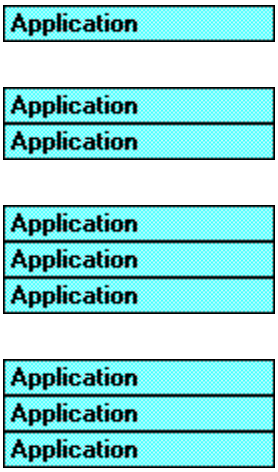
Using the PlaySettings Object

Use the **PlaySettings** property to return the **PlaySettings** object. The following example inserts a movie named "Clock.avi" into slide one in the active presentation, sets it to be played automatically after the previous animation or slide transition, specifies that the slide show continue while the movie plays, and specifies that the movie object be hidden during a slide show except when it's playing.

```
Set clockMovie = ActivePresentation.Slides(1).Shapes.AddMediaObject _
    FileName:="C:\WINNT\clock.avi", Left:=20, Top:=20
With clockMovie.AnimationSettings.PlaySettings
    .PlayOnEntry = True
    .PauseAnimation = False
    .HideWhileNotPlaying = True
End With
```

SlideShowSettings Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideShowSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideShowSettingsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideShowSettingsM"}



Represents the slide show setup for a presentation.

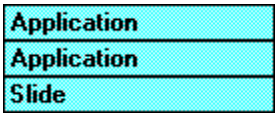
Using the SlideShowSettings Object

Use the **SlideShowSettings** property to return the **SlideShowSettings** object. The first section in the following example sets all the slides in the active presentation to advance automatically after five seconds. The second section sets the slide show to start on slide two, end on slide four, advance slides by using the timings set in the first section, and run in a continous loop until the user presses ESC. Finally, the example runs the slide show.

```
For Each s In ActivePresentation.Slides
    With s.SlideShowTransition
        .AdvanceOnTime = True
        .AdvanceTime = 5
    End With
Next

With ActivePresentation.SlideShowSettings
    .StartingSlide = 2
    .EndingSlide = 4
    .AdvanceMode = ppSlideShowUseSlideTimings
    .LoopUntilStopped = True
    .Run
End With
```

SlideShowView Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideShowViewC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideShowViewP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideShowViewM"}



Represents the view in a slide show window.

Using the SlideShowView Object

Use the **View** property of the **SlideShowWindow** object to return the **SlideShowView** object. The following example sets slide show window one to display the first slide in the presentation.

```
SlideShowWindows(1).View.First
```

Use the **Run** method of the **SlideShowSettings** object to create a **SlideShowWindow** object, and then use the **View** property to return the **SlideShowView** object the window contains. The following example runs a slide show of the active presentation, changes the pointer to a pen, and sets the pen color for the slide show to red.

```
With ActivePresentation.SlideShowSettings.Run.View
    .PointerColor.RGB = RGB(255, 0, 0)
    .PointerType = ppSlideShowPointerPen
End With
```

view

A way of displaying the contents of a presentation and providing the user the means to interact with it.

Tags Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTagsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTagsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTagsM"}

Represents a tag or a custom property that you can create for a shape, slide, or presentation. Each **Tags** object contains the name of a custom property and a value for that property.

Create tags when you want to be able to selectively work with specific members of a collection, based on an attribute that isn't already represented by a built-in property. For example, if you want to be able to categorize slides in a presentation based on what region of the country they apply to, you could create a Region tag and assign a Region value to each slide in the presentation. You could then selectively perform an operation on some of the slides, based on the values of their Region tags, such as hiding all the slides with the Region value "East."

Using the Tags Object

Use the **Add** method to add a tag to an object. The following example adds a tag with the name "Region" and with the value "East" to slide one in the active presentation.

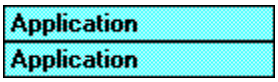
```
ActivePresentation.Slides(1).Tags.Add "Region", "East"
```

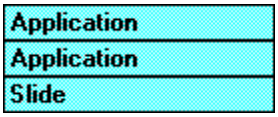
Use **Tags**(*index*), where *index* is the name of a tag, to return a the tag value. The following example tests the the value of the Region tag for all slides in the active presentation and hides any slides that don't pertain to the East Coast (denoted by the value "East").

```
For Each s In ActivePresentation.Slides
    If s.Tags("region") <> "east" Then
        s.SlideShowTransition.Hidden = True
    End If
Next
```

View Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideSorterViewC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideSorterViewP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideSorterViewM"}





Represents the current editing view in the specified document window.

Using the View Object

Use the **View** property of the **DocumentWindow** object to return the **View** object. The following example sets the size of window one and then sets the zoom to fit the new window size.

```
With Windows(1)
    .Height = 200
    .Width = 250
    .View.ZoomToFit = True
End With
```

Note The **View** object can represent any of the document window views: slide view, outline view, slide sorter view, notes page view, slide master view, handout master view, or notes master view. Some properties and methods of the **View** object work only in certain views. If you try to use a property or method that's inappropriate for a **View** object, an error occurs.

RGBColor Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjRGBColorC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjRGBColorP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjRGBColorM"}

Application

Application
Application

Represents a single color in a color scheme.

Using the RGBColor Object

Use the **Colors** method to return an **RGBColor** object. You can set an **RGBColor** object to another **RGBColor** object, or you can use the **RGB** property to set or return the explicit red-green-blue value for an **RGBColor** object. The following example sets the background color in color scheme one in the active presentation to red and sets the title color to the title color that's defined for color scheme two.

```
With ActivePresentation.ColorSchemes
    .Item(1).Colors(ppBackground).RGB = RGB(255, 0, 0)
    .Item(1).Colors(ppTitle) = .Item(2).Colors(ppTitle)
End With
```

Ruler Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjRulerC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjRulerP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjRulerM"}

Represents the ruler for the text in the specified shape or for all text in the specified text style. Contains tab stops and the indentation settings for text outline levels.

Using the Ruler Object

Use the **Ruler** property of the **TextFrame** object to return the **Ruler** object that represents the ruler for the text in the specified shape. Use the **TabStops** property to return the **TabStops** object that contains the tab stops on the ruler. Use the **Levels** property to return the **RulerLevels** object that contains the indentation settings for text outline levels. The following example sets a left-aligned tab stop at 2 inches (144 Points) and sets a hanging indent for the text in object two on slide one in the active presentation.

```
With ActivePresentation.Slides(1).Shapes(2).TextFrame.Ruler
    .TabStops.Add ppTabStopLeft, 144
    .Levels(1).FirstMargin = 0
    .Levels(1).LeftMargin = 36
End With
```

Use the **Ruler** property of the **TextStyle** object to return the **Ruler** object that represents the ruler for one of the four defined text styles (title text, body text, notes text, or default text). The followng example sets the first-line indent and hanging indent for outline level one in body text on the slide master for the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Ruler.Levels(1)
    .FirstMargin = 9
    .LeftMargin = 54
End With
```

RulerLevel Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjRulerLevelC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjRulerLevelP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjRulerLevelM"}

Contains first-line indent and hanging indent information for an outline level. The **RulerLevel** object is a member of the **RulerLevels** collection. The **RulerLevels** collection contains a **RulerLevel** object for each of the five available outline levels.

Using the RulerLevel Object

Use **RulerLevels**(*index*), where *index* is the outline level, to return a single **RulerLevel** object. The following example sets the first-line indent and hanging indent for outline level one in body text on the slide master for the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Ruler.Levels(1)
    .FirstMargin = 9
    .LeftMargin = 54
End With
```

The following example sets the first-line indent and hanging indent for outline level one in shape two on slide one in the active presentation.

```
With ActivePresentation.SlideMaster.Shapes(2).TextFrame.Ruler.Levels(1)
    .FirstMargin = 9
    .LeftMargin = 54
End With
```

RulerLevels Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjRulerLevelsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjRulerLevelsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjRulerLevelsM"}

A collection of all the **RulerLevel** objects on the specified ruler. Each **RulerLevel** object represents the first-line and left indent for text at a particular outline level. This collection always contains five members – one for each of the available outline levels.

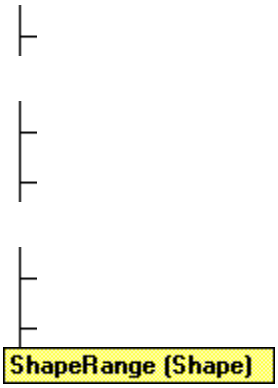
Using the RulerLevels Collection

Use the **Levels** property to return the **RulerLevels** collection. The following example sets the margins for the five outline levels in body text in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Ruler
    .Levels(1).FirstMargin = 0
    .Levels(1).LeftMargin = 40
    .Levels(2).FirstMargin = 60
    .Levels(2).LeftMargin = 100
    .Levels(3).FirstMargin = 120
    .Levels(3).LeftMargin = 160
    .Levels(4).FirstMargin = 180
    .Levels(4).LeftMargin = 220
    .Levels(5).FirstMargin = 240
    .Levels(5).LeftMargin = 280
End With
```

Selection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSelectionC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSelectionP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSelectionM"}



Represents the selection in the specified document window.

Using the Selection Object

Use the **Selection** property to return the **Selection** object. The following example places a copy of the selection in the active window on the Clipboard.

```
ActiveWindow.Selection.Copy
```

Use the **ShapeRange**, **SlideRange**, or **TextRange** property to return a range of shapes, slides, or text from the selection.

The following example sets the fill foreground color for the selected shapes in window two, assuming that there's at least one shape selected, and assuming that all selected shapes have a fill whose forecolor can be set.

```
With Windows(2).Selection.ShapeRange.Fill
    .Visible = True
    .ForeColor.RGB = RGB(255, 0, 255)
End With
```

The following example sets the text in the first selected shape in window two if that shape contains a text frame.

```
With Windows(2).Selection.ShapeRange(1)
    If .HasTextFrame Then
        .TextFrame.TextRange = "Current Choice"
    End If
End With
```

The following example cuts the selected text in the active window and places it on the Clipboard.

```
ActiveWindow.Selection.TextRange.Cut
```

The following example duplicates all the slides in the selection (if you're in slide view, this duplicates the current slide).

```
ActiveWindow.Selection.SlideRange.Duplicate
```

If you don't have an object of the appropriate type selected when you use one of these properties (for instance, if you use the **ShapeRange** property when there are no shapes selected), an error occurs. Use the **Type** property to determine what kind of object or objects are selected. The following example checks to see whether the selection contains slides. If the selection does contain slides, the example sets the background for the first slide in the selection.

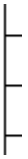
```
With Windows(2).Selection
    If .Type = ppSelectionSlides Then
        With .SlideRange(1)
            .FollowMasterBackground = False
            .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
        End With
    End If
End With
```

Remarks

The **Selection** object is deleted whenever you change slides in an active slide view (the **Type** property will return **ppSelectionNone**).

Slide Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideM"}



Represents a slide. The **Slide** object is a member of the **Slides** collection. The **Slides** collection contains all the **Slide** objects in a presentation.

Note Don't be confused if you're trying to return a reference to a single slide but you end up with a **SlideRange** object. A single slide can be represented either by a **Slide** object or by a **SlideRange** collection that contains only one slide, depending on how you return a reference to the slide. For example, if you create and return a reference to a slide by using the **Add** method, the slide is represented by a **Slide** object. However, if you create and return a reference to a slide by using the **Duplicate** method, the slide is represented by a **SlideRange** collection that contains a single slide. Because all the properties and methods that apply to a **Slide** object also apply to a **SlideRange** collection that contains a single slide, you can work with the returned slide in the same way, regardless of whether it's represented by a **Slide** object or a **SlideRange** collection.

Using the Slide Object

This section describes how to:

- Return a slide that you specify by name, index number, or slide ID number
- Return a slide in the selection
- Return the slide that's currently displayed in any document window or slide show window you specify
- Create a new slide

Returning a slide that you specify by name, index number, or slide ID number

Use **Slides**(*index*), where *index* is the slide name or index number, or use **Slides.FindBySlideID**(*index*), where *index* is the slide ID number, to return a single **Slide** object. The following example sets the layout for slide one in the active presentation.

```
ActivePresentation.Slides(1).Layout = ppLayoutTitle
```

The following example sets the layout for the slide with the ID number 265.

```
ActivePresentation.Slides.FindBySlideID(265).Layout = ppLayoutTitle
```

Returning a slide that you specify by name, index number, or slide ID number

Use **Slides**(*index*), where *index* is the slide name or index number, or use **Slides.FindBySlideID**(*index*), where *index* is the slide ID number, to return a single **Slide** object. The following example sets the layout for slide one in the active presentation.

```
ActivePresentation.Slides(1).Layout = ppLayoutTitle
```

Returning a slide in the selection

Use **Selection.SlideRange**(*index*), where *index* is the slide name or index number within the selection, to return a single **Slide** object. The following example sets the layout for slide one in the selection in the active window, assuming that there's at least one slide selected.

```
ActiveWindow.Selection.SlideRange(1).Layout = ppLayoutTitle
```

If there's only one slide selected, you can use **Selection.SlideRange** to return a **SlideRange** collection that contains the selected slide. The following example sets the layout for slide one in the current selection in the active window, assuming that there's exactly one slide selected.

```
ActiveWindow.Selection.SlideRange.Layout = ppLayoutTitle
```

Returning the slide that's currently displayed in any document window or slide show window you specify

Use the **Slide** property to return the slide that's currently displayed in the specified document window or slide show window view. The following example copies the slide that's currently displayed in document window two to the Clipboard.

```
Windows(2).View.Slide.Copy
```

Creating a new slide

Use the **Add** method to create a new slide and add it to the presentation. The following example adds a title slide to the beginning of the active presentation.

```
ActivePresentation.Slides.Add 1, ppLayoutTitleOnly
```

SlideRange Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideRangeM"}

A collection that represents a notes page or a slide range, which is a set of slides that can contain as little as a single slide or as much as all the slides in a presentation. You can include whichever slides you want – chosen from all the slides in the presentation or from all the slides in the selection – to construct a slide range. For example, you could construct a **SlideRange** collection that contains the first three slides in a presentation, all the selected slides in the presentation, or all the title slides in the presentation.

Using the SlideRange Collection

This section describes how to:

- Return a set of slides that you specify by name or index number
- Return all or some of the selected slides in a presentation
- Return a notes page
- Apply properties and methods to a slide range

Returning a set of slides that you specify by name or index number

Use **Slides.Range**(*index*), where *index* is the name or index number of the slide or an array that contains either names or index numbers of slides, to return a **SlideRange** collection that represents a set of slides in a presentation. You can use the **Array** function to construct an array of names or index numbers. The following example sets the background fill for slides one and three in the active presentation.

```
With ActivePresentation.Slides.Range(Array(1, 3))
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

The following example sets the background fill for the slides named "Intro" and "Big Chart" in the active presentation. Note that slides are assigned automatically generated names of the form Slide*n* (where n is an integer) when they're created. To assign a more meaningful name to a slide, use the **Name** property.

```
With ActivePresentation.Slides.Range(Array("Intro", "Big Chart"))
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

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

Returning all or some of the selected slides in a presentation

Use the **SlideRange** property of the **Selection** object to return all the slides in the selection. The following example sets the background fill for all the selected slides in window one, assuming that there's at least one slide selected.

```
With Windows(1).Selection.SlideRange
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

Use **Selection.SlideRange**(*index*), where *index* is the slide name or index number, to return a single slide from the selection. The following example sets the background fill for slide two in the collection of selected slides in window one, assuming that there are at least two slides selected.

```
With Windows(1).Selection.SlideRange(2)
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

Returning a notes page

Use the **NotesPage** property to return a **SlideRange** collection that represents the specified notes page. The following example inserts text into placeholder two (the notes area) on the notes page for slide one in the active presentation.

```
ActivePresentation.Slides(1).NotesPage.Shapes.Placeholders(2).TextFrame.Tex
tRange.InsertAfter "Added Text"
```

Applying a property or method to a slide range

Just as you can work with several slides at the same time in the user interface by selecting them and applying a command, you can work with several slides at the same time programmatically by constructing a **SlideRange** collection and applying properties or methods to it. And just as some commands in the user interface that work on single slides aren't valid when multiple slides are selected, some properties and methods that work on a **Slide** object or on a **SlideRange** collection that contains only one slide will fail if they're applied to a **SlideRange** collection that contains more than one slide. In general, if you can't do something manually when more than one slide is selected (such as return the individual shapes on one of the slides), you can't do it programmatically by using a **SlideRange** collection that contains more than one slide.

For those operations that work in the user interface whether you have a single slide or multiple slides selected (such as copying the selection to the Clipboard or setting the slide background fill), the associated properties and methods will work on a **SlideRange** collection that contains more than one slide. Here are some general guidelines for how these properties and methods behave when they're applied to multiple slides.

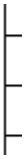
- Applying a method to a **SlideRange** collection is equivalent to applying the method to all the **Slide** objects in that range as a group.
- Setting the value of a property of the **SlideRange** collection is equivalent to setting the value of the property in each slide in that range individually (for a property that takes an enumerated type, setting the value to the "Mixed" value has no effect).
- A property of the **SlideRange** collection that returns an enumerated type returns the value of the property for an individual slide in the collection if all slides in the collection have the same value for

that property. If the slides in the collection don't all have the same value for the property, the property returns the "Mixed" value.

- A property of the **SlideRange** collection that returns a simple data type (such as **Long**, **Single**, or **String**) returns the value of the property for an individual slide in the collection if all slides in the collection have the same value for that property. If the slides in the collection don't all have the same value for the property, the property will return – 2 or generate an error. For example, using the **Name** property on a **SlideRange** object that contains multiple slides will generate an error because each slide has a different value for its **Name** property.
- Some formatting properties of slides aren't set by properties and methods that apply directly to the **SlideRange** collection, but by properties and methods that apply to an object contained in the **SlideRange** collection, such as the **ColorScheme** object. If the contained object represents operations that can be performed on multiple objects in the user interface, you'll be able to return the object from a **SlideRange** collection that contains more than one slide, and its properties and methods will follow the preceding rules. For example, you can use the **ColorScheme** property to return the **ColorScheme** object that represents the color schemes used on all the slides in the specified **SlideRange** collection. Setting properties for this **ColorScheme** object will also set these properties for the **ColorScheme** objects on all the individual slides in the **SlideRange** collection.

Slides Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlidesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlidesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlidesM"}



A collection of all the **Slide** objects in the specified presentation.

Using the Slides Collection

This section describes how to:

- Create a slide and add it to the collection
- Return a single slide that you specify by name, index number, or slide ID number
- Return a subset of the slides in the presentation
- Apply a property or method to all the slides in the presentation at the same time

Creating a slide and adding it to the collection

Use the **Slides** property to return a **Slides** collection. Use the **Add** method to create a new slide and add it to the collection. The following example adds a new slide to the active presentation.

```
ActivePresentation.Slides.Add 2, ppLayoutBlank
```

Returning a single slide that you specify by name, index number, or slide ID number

Use **Slides**(*index*), where *index* is the slide name or index number, or use the **Slides.FindBySlideID**(*index*), where *index* is the slide ID number, to return a single **Slide** object. The following example sets the layout for slide one in the active presentation.

```
ActivePresentation.Slides(1).Layout = ppLayoutTitle
```

The following example sets the layout for the slide named "Big Chart" in the active presentation. Note that slides are assigned automatically generated names of the form Slide*n* (where *n* is an integer) when they're created. To assign a more meaningful name to a slide, use the **Name** property.

```
ActivePresentation.Slides("Big Chart").Layout = ppLayoutTitle
```

Returning a subset of the slides in the presentation

Use **Slides.Range**(*index*), where *index* is the slide index number or name or an array of slide index numbers or an array of slide names, to return a **SlideRange** object that represents a subset of the **Slides** collection. The following example sets the background fill for slides one and three in the active presentation.

```
With ActivePresentation.Slides.Range(Array(1, 3))
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
```

```
End With
```

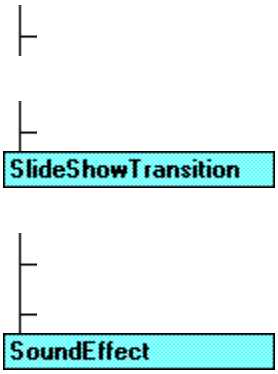
Applying a property or method to all the slides in the presentation at the same time

If you want to do something to all the slides in your presentation at the same time (such as delete all of them or set a property for all of them), use **Slides.Range** with no argument to construct a **SlideRange** collection that contains all the slides in the **Slides** collection, and then apply the appropriate property or method to the **SlideRange** collection. The following example sets the background fill for all the slides in the active presentation

```
With ActivePresentation.Slides.Range
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

SlideShowTransition Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideShowTransitionC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideShowTransitionP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideShowTransitionM"}



Contains information about how the specified slide advances during a slide show.

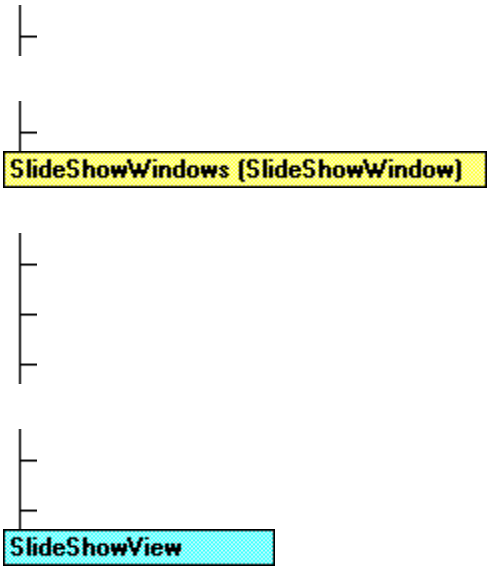
Using the SlideShowTransition Object

Use the **SlideShowTransition** property to return the **SlideShowTransition** object. The following example specifies a Fast Strips Down-Left transition accompanied by the Bass.wav sound for slide one in the active presentation and specifies that the slide advance automatically five seconds after the previous animation or slide transition.

```
With ActivePresentation.Slides(1).SlideShowTransition
    .Speed = ppTransitionSpeedFast
    .EntryEffect = ppEffectStripsDownLeft
    .SoundEffect.ImportFromFile "c:\sndsys\bass.wav"
    .AdvanceOnTime = True
    .AdvanceTime = 5
End With
ActivePresentation.SlideShowSettings.AdvanceMode = _
    ppSlideShowUseSlideTimings
```

SlideShowWindow Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideShowWindowC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideShowWindowP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideShowWindowM"}



Represents a window in which a slide show runs. The **SlideShowWindow** object is a member of the **SlideShowWindows** collection. The **SlideShowWindows** collection contains all the open slide show windows.

Using the SlideShowWindow Object

Use **SlideShowWindows**(*index*), where *index* is the slide show window index number, to return a single **SlideShowWindow** object. The following example activates slide show window two.

```
SlideShowWindows(2).Activate
```

Use the **Run** method to create a new slide show window and return a reference to this slide show window. The following example runs a slide show of the active presentation and reduces the height of the slide show window just enough so that you can see the taskbar (for screens with a resolution of 800 by 600).

```
With ActivePresentation.SlideShowSettings
    .ShowType = ppShowTypeSpeaker
    With .Run
        .Height = 300
        .Width = 400
    End With
End With
```

Use the **View** property to return the view in the specified slide show window. The following example sets the view in slide show window one to display slide three in the presentation.

```
SlideShowWindows(1).View.GotoSlide 3
```

Use the **Presentation** property to return the presentation that's currently running in the specified slide show window. The following example displays the name of the presentation that's currently running in slide show window one.

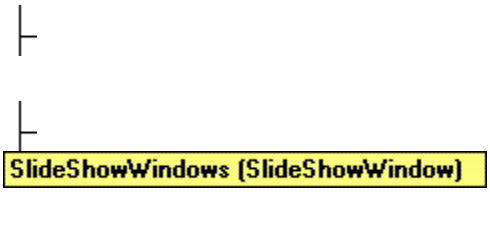
```
MsgBox SlideShowWindows(1).Presentation.Name
```

```
Dim p as Presentation
Dim w as SlideShowWindow
```

```
Set p = the presentation whose slide show window you want to find
For i = 1 to SlideShowWindows.Count
if( SlideShowWindows(i).Presentation = p )
   Set w = SlideShowWindows(i)
Next i
```

SlideShowWindows Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSlideShowWindowsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSlideShowWindowsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSlideShowWindowsM"}



A collection of all the **SlideShowWindow** objects that represent the open slide shows in PowerPoint.

Using the SlideShowWindows Collection

Use the **SlideShowWindows** property to return the **SlideShowWindows** collection. Use **SlideShowWindows**(*index*), where *index* is the window index number, to return a single **SlideShowWindow** object. The following example reduces the height of slide show window one if it's a full-screen window.

```
With SlideShowWindows(1)
    If .IsFullScreen Then
        .Height = .Height - 20
    End If
End With
```

Use the **Run** method to create a new slide show window and add it to the **SlideShowWindows** collection. The following example runs a slide show of the active presentation.

```
ActivePresentation.SlideShowSettings.Run
```

SoundEffect Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjSoundEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjSoundEffectP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjSoundEffectM"}

Represents the sound effect that accompanies an animation or slide transition in a slide show.

Using the SoundEffect Object

Use the **SoundEffect** property of the **AnimationSettings** object to return the **SoundEffect** object that represents the sound effect that accompanies an animation. The following example specifies that the animation of the title on slide one in the active presentation be accompanied by the sound in the Bass.wav file.

```
With ActivePresentation.Slides(1).Shapes(1).AnimationSettings
    .TextLevelEffect = ppAnimateByAllLevels
    .SoundEffect.ImportFromFile "c:\sndsys\bass.wav"
End With
```

Use the **SoundEffect** property of the **SlideShowTransition** object to return the **SoundEffect** object that represents the sound effect that accompanies a slide transition.

The following example specifies that the transition to slide one in the active presentation be accompanied by the sound in the Bass.wav file.

```
ActivePresentation.Slides(1).SlideShowTransition.SoundEffect _
    .ImportFromFile "c:\sndsys\bass.wav"
```

TabStop Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTabStopC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTabStopP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTabStopM"}

Represents a single tab stop. The **TabStop** object is a member of the **TabStops** collection. The **TabStops** collection represents all the tab stops on one ruler.

Using The Tabstop Object

Use **TabStops**(*index*), where *index* is the tab stop index number, to return a single **TabStop** object. The following example clears tab stop one for the text in shape two on slide one in the active presentation.

```
ActivePresentation.Slides(1).Shapes(2).TextFrame.Ruler.TabStops(1).Clear
```

TabStops Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTabStopsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTabStopsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTabStopsM"}

A collection of all the **TabStop** objects on one ruler.

Using the TabStops Collection

Use the **TabStops** property to return the **TabStops** collection. The following example clears all the tab stops for the text in shape two on slide one in the active presentation.

```
With ActivePresentation.Slides(1).Shapes(2).TextFrame.Ruler.TabStops
    For t = .Count To 1 Step -1
        .Item(t).Clear
    Next
End With
```

Use the **Add** method to create a tab stop and add it to the **TabStops** collection. The following example adds a tab stop to the body-text style on the slide master for the active presentation. The new tab stop will be positioned 2 inches (144 points) from the left edge of the ruler and will be left aligned.

```
ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Ruler.TabStops.Add
ppTabStopLeft, 144
```

TextStyle Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextStyleC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextStyleP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextStyleM"}

Represents one of three text styles: title text, body text, or default text. Each text style contains a **TextFrame** object that describes how text is placed within the text bounding box, a **Ruler** object that contains tab stops and outline indent formatting information, and a **TextStyleLevels** collection that contains outline text formatting information. The **TextStyle** object is a member of the **TextStyles** collection.

Using the TextStyle Object

Use **TextStyles**(*index*), where *index* is either **ppBodyStyle**, **ppDefaultStyle**, or **ppTitleStyle**, to return a single **TextStyle** object. The following example sets the font name and font size for level-one body text on all the slides in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Levels(1)
    With .Font
        .Name = "Arial"
        .Size = 36
    End With
End With
```

TextStyles Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextStylesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextStylesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextStylesM"}

A collection of three text styles – title text, body text, and default text

– each of which is represented by a **TextStyle** object. Each text style contains a **TextFrame** object that describes how text is placed within the text bounding box, a **Ruler** object that contains tab stops and outline indent formatting information, and a **TextStyleLevels** collection that contains outline text formatting information.

Using the TextStyles Collection

Use **TextStyles**(*index*), where *index* is either **ppBodyStyle**, **ppDefaultStyle**, or **ppTitleStyle**, to return a single **TextStyle** object. This example sets the margins for the notes body area on all the notes pages in the active presentation.

```
With ActivePresentation.NotesMaster.TextStyles(ppBodyStyle).TextFrame
    .MarginBottom = 50
    .MarginLeft = 50
    .MarginRight = 50
    .MarginTop = 50
End With
```

TextStyleLevel Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextStyleLevelC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextStyleLevelP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextStyleLevelM"}

TextStyleLevels (TextStyleLevel)

Contains character and paragraph formatting information for an outline level. The **TextStyleLevel** object is a member of the **TextStyleLevels** collection. The **TextStyleLevels** collection contains one **TextStyleLevel** object for each of the five outline levels

Using the TextStyleLevel Object

Use **Levels**(*index*), where *index* is a number from 1 through 5 that corresponds to the outline level, to return a single **TextStyleLevel** object. The following example sets the font name and font size, the space before paragraphs, and the paragraph alignment for level-one body text on all the slides in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Levels(1)
    With .Font
        .Name = "Arial"
        .Size = 36
    End With
    With .ParagraphFormat
        .LineRuleBefore = False
        .SpaceBefore = 14
        .Alignment = ppAlignJustify
    End With
End With
```

TextStyleLevels Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextStyleLevelsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextStyleLevelsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextStyleLevelsM"}

TextStyleLevels (TextStyleLevel)

A collection of all the outline text levels. This collection always contains five members, each of which is represented by a **TextStyleLevel** object.

Using the TextStyleLevels Collection

Use **Levels**(*index*), where *index* is a number from 1 through 5 that corresponds to the outline level, to return a single **TextStyleLevel** object. The following example sets the font name and font size for level-one body text on all the slides in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Levels(1)
    With .Font
        .Name = "Arial"
        .Size = 36
    End With
End With
```

The following example sets the font size for text at each outline level for the notes body area on all the notes pages in the active presentation.

```
With ActivePresentation.NotesMaster.TextStyles(ppBodyStyle).Levels
    .Item(1).Font.Size = 34
    .Item(2).Font.Size = 30
    .Item(3).Font.Size = 25
    .Item(4).Font.Size = 20
    .Item(5).Font.Size = 15
End With
```

Adjustments Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjAdjustmentsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjAdjustmentsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjAdjustmentsM"}

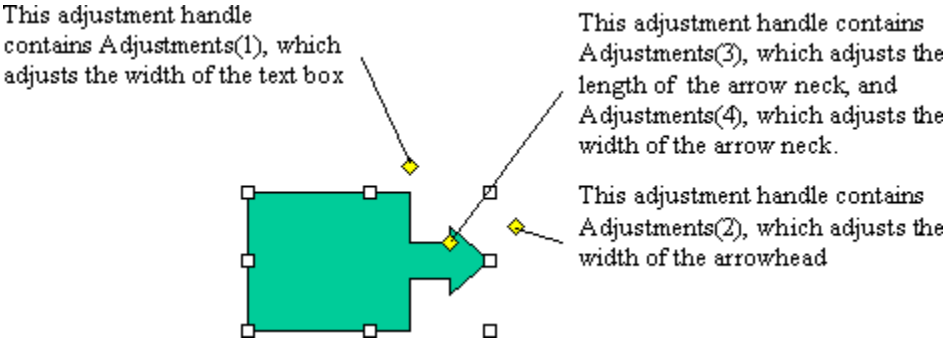
Adjustments

Contains a collection of adjustment values for the specified AutoShape, WordArt object, or connector. 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. For example, the following illustration shows what each of the four adjustment values for a right-arrow callout contributes to the definition of the callout's geometry.



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 connectors and callouts, where the values 0.0 and 1.0 represent the limits of the rectangle defined by

	the starting and ending points of the connector or 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 `myDocument` 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 myDocument = ActivePresentation.Slides(1)
Set rac = myDocument.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
```

CalloutFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjCalloutFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjCalloutFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjCalloutFormatM"}

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 `myDocument`: 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Callout
    .Accent = True
    .Angle = msoCalloutAngle30
    .Border = False
    .PresetDrop msoCalloutDropTop
    .Type = msoCalloutThree
End With
```

ColorFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjColorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjColorFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjColorFormatM"}

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

Using the ColorFormat Object

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

To return a ColorFormat object that represents this	Use this property	With this object
Color used for dimmed objects	**DimColor**	**AnimationSettings**
Background fill color (used in a shaded or patterned fill)	**BackColor**	**FillFormat**
Foreground fill color (or simply the fill color for a solid fill)	**ForeColor**	**FillFormat**
Bullet or character color	**Color**	**Font**
Background line color (used in a patterned line)	**BackColor**	**LineFormat**
Foreground line color (or just the line color for a solid line)	**ForeColor**	**LineFormat**
Shadow color	**ForeColor**	**ShadowFormat**
Default pointer color for a presentation	**PointerColor**	**SlideShowSettings**
Temporary pointer color for a view of a slide show	**PointerColor**	**SlideShowView**
Color of the sides of an extruded object	**ExtrusionColor**	**ThreeDFormat**

Use the **SchemeColor** property to set the color of a slide element to one of the colors in the standard color scheme. The following example sets the text color for shape one on slide two in the active presentation to the standard color-scheme title color.

```
ActivePresentation.Slides(2).Shapes(1).TextFrame.TextRange.Font.Color.Schem
eColor = ppTitle
```

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

```
Set myDocument = ActivePresentation.Slides(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
```

ConnectorFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjConnectorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjConnectorFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjConnectorFormatM"}

ConnectorFormat

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

Using the ConnectorFormat Object

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

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

The following example adds two rectangles to `myDocument` and connects them with a curved connector.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 0, 0).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    .Parent.RerouteConnections
End With
```

Remarks

Connection sites are generally numbered according to the rules presented in the following table.

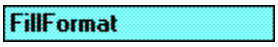
Shape type	Connection site numbering scheme
AutoShapes, WordArt, pictures, and OLE objects	The connection sites are numbered starting at the top and proceeding counterclockwise.
Freeforms	The connection sites are the vertices, and they correspond to the vertex numbers.

To figure out which number corresponds to which connection site on a complex shape, you can experiment with the shape while the macro recorder is turned on and then examine the recorded code; or you can create a shape, select it, and then run the following example. This code will number each connection site and attach a connector to it.

```
Set mainshape = ActiveWindow.Selection.ShapeRange(1)
With mainshape
    bx = .Left + .Width + 50
    by = .Top + .Height + 50
End With
With ActiveWindow.View.Slide
    For j = 1 To mainshape.ConnectionSiteCount
        With .Shapes.AddConnector(msoConnectorStraight, bx, by, bx + 50, by
+ 50)
            .ConnectorFormat.EndConnect mainshape, j
            .ConnectorFormat.Type = msoConnectorElbow
            .Line.ForeColor.RGB = RGB(255, 0, 0)
            l = .Left
            t = .Top
        End With
        With .Shapes.AddTextbox(msoTextOrientationHorizontal, l, t, 36, 14)
            .Fill.Visible = False
            .Line.Visible = False
            .TextFrame.TextRange.Text = j
        End With
    Next j
End With
```

FillFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjFillFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjFillFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjFillFormatM"}



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 `myDocument` and then sets the gradient and color for the rectangle's fill.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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.

FreeformBuilder Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjFreeformBuilderC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjFreeformBuilderP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjFreeformBuilderM"}

FreeformBuilder

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 freefrom. 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 `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

GroupShapes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjGroupShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjGroupShapesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjGroupShapesM"}

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 the **GroupShapes** collection. The following 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 = ActivePresentation.Slides(1)
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
```

LineFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjLineFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjLineFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjLineFormatM"}

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 a blue, dashed line to `myDocument`. There's a short, narrow oval at the line's starting point and a long, wide triangle at its end point.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

LinkFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjLinkFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjLinkFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjLinkFormatM"}

LinkFormat

Contains properties and methods that apply to linked OLE objects. The **OLEFormat** object contains properties and methods that apply to OLE objects whether or not they're linked. The **PictureFormat** object contains properties and methods that apply to pictures and OLE objects.

Using the LinkFormat Object

Use the **LinkFormat** property to return a **LinkFormat** object. The following example loops through all the shapes on all the slides in the active presentation and sets all linked Microsoft Excel worksheets to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Excel.Sheet.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

ObjectVerbs Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjObjectVerbsC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjObjectVerbsP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjObjectVerbsM"}

OLEFormat

ObjectVerbs

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

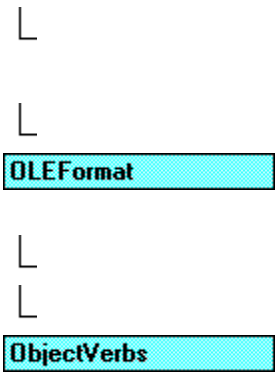
Using the ObjectVerbs Object

Use the **ObjectVerbs** property to return an **ObjectVerbs** object. The following example displays all the available verbs for the OLE object contained in shape one on slide two in the active presentation. For this example to work, shape one must contain an OLE object.

```
With ActivePresentation.Slides(2).Shapes(1).OLEFormat
    For Each v In .ObjectVerbs
        MsgBox v
    Next
End With
```

OLEFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjOLEFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjOLEFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjOLEFormatM"}



Contains properties and methods that apply to OLE objects. The **LinkFormat** object contains properties and methods that apply to linked OLE objects only. The **PictureFormat** object contains properties and methods that apply to pictures and OLE objects.

Using the OLEFormat Object

Use the **OLEFormat** property to return an **OLEFormat** object. The following example loops through all the shapes on all the slides in the active presentation and sets all linked Microsoft Excel worksheets to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Excel.Sheet.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

PictureFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjPictureFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjPictureFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjPictureFormatM"}

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 `myDocument` 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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = 0.7
    .ColorType = msoPictureGrayScale
    .CropBottom = 18
End With
```

ShadowFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShadowFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShadowFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShadowFormatM"}

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 `myDocument`. The semitransparent, blue shadow is offset 5 points to the right of the rectangle and 3 points above it.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShapeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShapeM"}

Represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, or picture. The **Shape** object is a member of the **Shapes** collection. The **Shapes** collection contains all the shapes on a slide.

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. For an overview of how to work with either a single shape or with more than one shape at a time, see Working with Shapes (Drawing Objects).

Using the Shape Object

This section describes how to:

- Return an existing shape on a slide, indexed by name or number.
- Return a newly created shape on a slide.
- Return a shape within the selection.
- Return the slide title and other placeholders on a slide.
- Return the shapes attached to the ends of a connector.
- Return the default shape for a presentation.
- Return a newly created freeform.
- Return a single shape from within a group.
- Return a newly formed group of shapes.

Returning an Existing Shape on a Slide

Use **Shapes**(*index*), where *index* is the shape name or the index number, to return a **Shape** object that represents a shape on a slide. The following example horizontally flips shape one and the shape named "Rectangle 1" on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).Flip msoFlipHorizontal
myDocument.Shapes("Rectangle 1").Flip msoFlipHorizontal
```

Each shape is assigned a default name when you add it to the **Shapes** collection. To give the shape a more meaningful name, use the **Name** property. The following example adds a rectangle to

`myDocument`, gives it the name "Red Square," and then sets its foreground color and line style.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 144, 144, 72, 72)
    .Name = "Red Square"
    .Fill.ForeColor.RGB = RGB(255, 0, 0)
    .Line.DashStyle = msoLineDashDot
End With
```

Returning a Newly Created Shape on a Slide

To add a shape to a slide and return a **Shape** object that represents the newly created shape, use one of the following methods of the **Shapes** collection: **AddCallout**, **AddComment**, **AddConnector**, **AddCurve**, **AddLabel**, **AddLine**, **AddMediaObject**, **AddOLEObject**, **AddPicture**, **AddPlaceholder**, **AddPolyline**, **AddShape**, **AddTextbox**, **AddTextEffect**, **AddTitle**.

Returning a Shape Within the Selection

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

```
ActiveWindow.Selection.ShapeRange(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
```

Returning the Slide Title and Other Placeholders on a Slide

Use **Shapes.Title** to return a **Shape** object that represents an existing slide title. Use **Shapes.AddTitle** to add a title to a slide that doesn't already have one and return a **Shape** object that represents the newly created title. Use **Shapes.Placeholders**(*index*), where *index* is the placeholder's index number, to return a **Shape** object that represents a placeholder. If you have not changed the layering order of the shapes on a slide, the following three statements are equivalent, assuming that slide one has a title.

```
ActivePresentation.Slides(1).Shapes.Title.TextFrame.TextRange.Font.Italic =
True
ActivePresentation.Slides(1).Shapes.Placeholders(1).TextFrame.TextRange.Fon
t.Italic = True
ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange.Font.Italic =
True
```

Returning the Shapes Attached to the Ends of a Connector

To return a **Shape** object that represents one of the shapes attached by a connector, use the **BeginConnectedShape** or **EndConnectedShape** property.

Returning the Default Shape for a Presentation

To return a **Shape** object that represents the default shape for a presentation, use the **DefaultShape** property.

Returning a newly created freeform

Use the **BuildFreeform** and **AddNodes** methods to define the geometry of a new freeform, and use the **ConvertToShape** method to create the freeform and return the **Shape** object that represents it.

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.

ShapeNode Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShapeNodeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShapeNodeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShapeNodeM"}

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 `myDocument` is a corner point, the following example makes it a smooth point. For this example to work, shape three must be a freeform.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Nodes(1).EditingType = msoEditingCorner Then
        .Nodes.SetEditingType 1, msoEditingSmooth
    End If
End With
```

ShapeNodes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShapeNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShapeNodesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShapeNodesM"}

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 `myDocument`. For this example to work, shape three must be a freeform with at least four nodes.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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 `myDocument`. For this example to work, shape three must be a freeform with at least four nodes.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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 `myDocument` is a corner point, the following example makes it a smooth point. For this example to work, shape three must be a freeform.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Nodes(1).EditingType = msoEditingCorner Then
        .Nodes.SetEditingType 1, msoEditingSmooth
    End If
End With
```

ShapeRange Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShapeRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShapeRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShapeRangeM"}

Represents a shape range, which is a set of shapes on a document. A shape range can contain as few as a single shape or as many as all the shapes on the document. You can include whichever shapes you want – chosen from among all the shapes on 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 on a document, all the selected shapes on a document, or all the freeforms on a document.

For an overview of how to work with either a single shape or with more than one shape at a time, see Working with Shapes (Drawing Objects).

Using the ShapeRange Collection

This section describes how to:

- Return a set of shapes you specify by name or index number.
- Return all or some of the selected shapes on a document.

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 the **Array** function to construct an array of names or index numbers. The following example sets the fill pattern for shapes one and three on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned
msoPatternHorizontalBrick
```

The following example sets the fill pattern for the shapes named "Oval 4" and "Rectangle 5" on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
Set myRange = myDocument.Shapes.Range(Array("Oval 4", "Rectangle 5"))
myRange.Fill.Patterned msoPatternHorizontalBrick
```

Although you can use the **Range** method to return any number of shapes or slides, 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 All or Some of the Selected Shapes on a Document

Use the **ShapeRange** property of the **Selection** object to return all the shapes in the selection. The following example sets the fill foreground color for all the shapes in the selection in window one, assuming that there's at least one shape in the selection.

```
Windows(1).Selection.ShapeRange.Fill.ForeColor.RGB = RGB(255, 0, 255)
```

Use **Selection.ShapeRange**(*index*), where *index* is the shape name or the index number, to return a single shape within the selection. The following example sets the fill foreground color for shape two in the collection of selected shapes in window one, assuming that there are at least two shapes in the selection.

```
Windows(1).Selection.ShapeRange(2).Fill.ForeColor.RGB = RGB(255, 0, 255)
```

Shapes Collection Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjShapesP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjShapesM"}

A collection of all the **Shape** objects on the specified slide. 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. For an overview of how to work either with a single shape or with more than one shape at a time, see Working with Shapes (Drawing Objects).

Using the Shapes Collection

Use the **Shapes** property to return the **Shapes** collection.The following example selects all the shapes on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.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 with no argument 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 the **AddCallout**, **AddComment**, **AddConnector**, **AddCurve**, **AddLabel**, **AddLine**, **AddMediaObject**, **AddOLEObject**, **AddPicture**, **AddPlaceholder**, **AddPolyline**, **AddShape**, **AddTextbox**, **AddTextEffect**, or **AddTitle** method to create a new shape and add it to the **Shapes** collection. Use the **BuildFreeform** method in conjunction with the **ConvertToShape** method to create a new freeform and add it to the collection. The following example adds a rectangle to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape msoShapeRectangle, 50, 50, 100, 200
```

Use **Shapes**(*index*), where *index* is the shape's name or index number, to return a single **Shape** object. The following example sets the fill to a preset shade for shape one on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientBrass
```

Use **Shapes.Range**(*index*), where *index* is the shape's name or index number or an array of shape names or index numbers, to return a **ShapeRange** collection that represents a subset of the **Shapes** collection. The following example sets the fill pattern for shapes one and three on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned
msoPatternHorizontalBrick
```

Use **Shapes.Placeholders**(*index*), where *index* is the placeholder number, to return a **Shape** object that represents a placeholder. If the specified slide has a title, use **Shapes.Placeholders(1)** or **Shapes.Title** to return the title placeholder. The following example adds a slide to the active presentation and then adds text to both the title and the subtitle (the subtitle is the second placeholder on a slide with this layout).

```
Set myDocument = ActivePresentation.Slides(1)
With ActivePresentation.Slides.Add(1, ppLayoutTitle).Shapes
    .Title.TextFrame.TextRange = "This is the title text"
    .Placeholders(2).TextFrame.TextRange = "This is subtitle text"
End With
```

TextEffectFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextEffectFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextEffectFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextEffectFormatM"}

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 `myDocument`. For this example to work, shape one must be a WordArt object.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).TextEffect
    .FontName = "Courier New"
    .FontBold = True
    .FontItalic = True
End With
```

TextFrame Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextFrameC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextFrameP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextFrameM"}

Represents the text frame in a **Shape** object. Contains the text in the text frame as well as the properties and methods that control the alignment and anchoring of the text frame.

Using the TextFrame Object

Use the **TextFrame** property to return a **TextFrame** object. The following example adds a rectangle to `myDocument`, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame
    .TextRange.Text = "Here is some test text"
    .MarginBottom = 10
    .MarginLeft = 10
    .MarginRight = 10
    .MarginTop = 10
End With
```

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

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.HasTextFrame Then
        With s.TextFrame
            If .HasText Then MsgBox .TextRange.Text
        End With
    End If
Next
```

text frame

The area within a shape that can contain text.

TextRange Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjTextRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjTextRangeP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjTextRangeM"}

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

Using the TextRange Object

This section describes how to:

- Return the text range in any shape you specify.
- Return a text range from the selection.
- Return particular characters, words, lines, sentences, or paragraphs from a text range.
- Find and replace text in a text range.
- Insert text, the date and time, or the slide number into a text range.
- Position the insertion point wherever you want in a text range.

Returning a Text Range from Any Shape You Specify

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

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 250,
140).TextFrame.TextRange.Text = "Here is some test text"
```

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

```
ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange.Text = "Here is
some test text"
ActivePresentation.Slides(1).Shapes(1).TextFrame.TextRange = "Here is some
test text"
```

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

Returning a Text Range from the Selection

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

```
ActiveWindow.Selection.TextRange.Copy
```

Returning Particular Characters, Words, Lines, Sentences, or Paragraphs from a Text Range

Use one of the following methods to return a portion of the text of a **TextRange** object: **Characters**, **Lines**, **Paragraphs**, **Runs**, **Sentences**, or **Words**.

Finding and Replacing Text in a Text Range

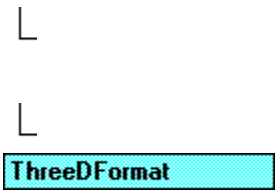
Use the **Find** and **Replace** methods to find and replace text in a text range.

Inserting Text, the Date and Time, or the Slide Number into a Text Range

Use one of the following methods to insert characters into a **TextRange** object: **InsertAfter**, **InsertBefore**, **InsertDateTime**, **InsertSlideNumber**, or **InsertSymbol**.

ThreeDFormat Object

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppobjThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Properties":"ppobjThreeDFormatP"} {ewc HLP95EN.DLL, DYNALINK, "Methods":"ppobjThreeDFormatM"}



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 `myDocument` and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Set myDocument = ActivePresentation.Slides(1)
Set myShape = myDocument.Shapes.AddShape(msoShapeOval, 90, 90, 90, 40)
With myShape.ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)    ' RGB value for purple
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.

Shape, **ShapeRange**, **TextRange**

AddIns, **Assistant**, **CommandBars**,**DocumentWindows**, **FileFind**, **FileSearch**, **Presentation**, **Presentations**, **SlideShowWindows**, **VBE**

AnimationSettings, **FillFormat**, **Font**, **LineFormat**, **ShadowFormat**, **SlideShowSettings**, **SlideShowView**, **ThreeDFormat**

ColorSchemes, **Master**, **Slide**, **SlideRange**

Application, **Presentation**

BulletFormat, **Fonts**, **TextRange**, **TextStyleLevel**

ActionSetting, **Hyperlinks**

Presentation, **Slide**, **SlideRange**

TextRange, **TextStyleLevel**

ColorSchemes, **DocumentProperties**, **DocumentWindows**, **ExtraColors**, **Fonts**, **Master**, **PageSetup**, **PrintOptions**, **Slides**, **SlideShowSettings**, **Tags**, **VBProject**

Application, **DocumentWindow**, **Presentations**, **SlideShowWindow**

TextFrame, **TextStyle**

ActionSettings, **Adjustments**, **AnimationSettings**, **CalloutFormat**, **ConnectorFormat**, **FillFormat**, **GroupShapes**, **LineFormat**, **LinkFormat**, **OLEFormat**, **PictureFormat**, **PlaceholderFormat**, **ShadowFormat**, **ShapeNodes**, **ShapeRange**, **Tags**, **TextEffectFormat**, **TextFrame**, **ThreeDFormat**

ActionSettings, **Adjustments**, **AnimationSettings**, **CalloutFormat**, **ConnectorFormat**, **FillFormat**, **GroupShapes**, **LineFormat**, **LinkFormat**, **OLEFormat**, **PictureFormat**, **PlaceholderFormat**, **ShadowFormat**, **Shape**, **ShapeNodes**, **ShapeRange**, **Tags**, **TextEffectFormat**, **TextFrame**, **ThreeDFormat**

Master, **Selection**, **Shapes**, **Slide**, **SlideRange**

ColorScheme, **HeadersFooters**, **Hyperlinks**, **ShapeRange**, **Shapes**, **SlideShowTransition**, **Tags**

ColorScheme, **HeadersFooters**, **Hyperlinks**, **ShapeRange**, **Shapes**, **SlideShowTransition**, **Tags**

Selection, **Slides**

ActionSettings, **AnimationSettings**, **SlideShowTransition**

Presentation, **Shape**, **ShapeRange**, **Slide**, **SlideRange**

Shapes, **TextStyles**

Selection, **TextFrame**, **TextRange**

AutoUpdate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoUpdateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoUpdateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoUpdateA"}

Returns or sets the way the link will be updated automatically (each time the presentation is opened or the source file changes) or manually (only when the user specifically asks to update the presentation). Can be one of the following **PpUpdateOption** constants: **ppUpdateOptionAutomatic**, **ppUpdateOptionManual**, or **ppUpdateOptionMixed**. Read/write **Long**.

AutoUpdate Property Example

This example loops through all the shapes on all the slides in the active presentation and sets all linked Microsoft Excel worksheets to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Excel.Sheet.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

SourceFullName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSourceFullNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSourceFullNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSourceFullNameA"}

Returns or sets the name and path of the source file for the linked OLE object. Read/write **String**.

SourceFullName Property Example

This example sets the source file for shape one on slide one in the active presentation to Wordtest.doc and specifies that the object's image be updated automatically.

```
With ActivePresentation.Slides(1).Shapes(1)
    If .Type = msoLinkedOLEObject Then
        With .LinkFormat
            .SourceFullName = "c:\my documents\wordtest.doc"
            .AutoUpdate = ppUpdateOptionAutomatic
        End With
    End If
End With
```

Update Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUpdateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUpdateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUpdateA"}

Updates the specified linked OLE object. To update all the links in a presentation at once, use the **UpdateLinks** method.

Syntax

expression.**Update**

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

Update Method Example

This example updates all linked OLE objects in the active presentation.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            sh.LinkFormat.Update
        End If
    Next
Next
```

DoVerb Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDoVerbC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDoVerbX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDoVerbA"}

Requests that an OLE object perform one of its verbs. Use the **ObjectVerbs** property to determine the available verbs for an OLE object.

Syntax

expression.**DoVerb(*Index*)**

expression Required. An expression that returns an **OLEFormat** object.

Index Optional **Integer**. The verb to perform. If this argument is omitted, the default verb is performed.

DoVerb Method Example

This example performs the default verb for shape three on slide one in the active presentation if shape three is alinked or embedded OLE object.

```
With ActivePresentation.Slides(1).Shapes(3)
    If .Type = msoEmbeddedOLEObject Or _
            .Type = msoLinkedOLEObject Then
        .OLEFormat.DoVerb
    End If
End With
```

This example performs the verb "Open" for shape three on slide one in the active presentation if shape three is an OLE object that supports the verb "Play".

```
With ActivePresentation.Slides(1).Shapes(3)
    If .Type = msoEmbeddedOLEObject Or _
            .Type = msoLinkedOLEObject Then
        For Each sVerb In .OLEFormat.ObjectVerbs
            nCount = nCount + 1
            If sVerb = "Open" Then
                .OLEFormat.DoVerb nCount
                Exit For
            End If
        Next
    End If
End With
```

verb

An action that an OLE object takes to activate its content. Common verbs are "Edit", "Open", and "Play."

FollowColors Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro FollowColorsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro FollowColorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro FollowColorsA"}

Returns or sets the extent to which the colors in the specified object follow the slide's color scheme. The specified object must be a chart created in either Microsoft Graph or Microsoft Organization Chart. The property can have one of the following **PpFollowColors** constants: **ppFollowColorsScheme**, if all the colors in the chart follow the slide's color scheme; **ppFollowColorsTextAndBackground**, if only the text and background follow the slide's color scheme; or **ppFollowColorsNone**, if the chart colors don't follow the slide's color scheme. Read/write **Long**.

FollowColors Property Example

This example specifies that the text and background of shape two on slide one in the active presentation follow the slide's color scheme. Shape two must be a chart created in either Microsoft Graph or Microsoft Organization Chart.

```
ActivePresentation.Slides(1).Shapes(2).OLEFormat.FollowColors = _
    ppFollowColorsTextAndBackground
```

Object Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproObjectA"}

Returns the object that represents the specified OLE object's top-level interface. This property allows you to access the properties and methods of the application in which an OLE object was created. Read-only.

Remarks

Use the **TypeName** function to determine the type of object this property returns for a specific OLE object.

Object Property Example

This example displays the type of object contained in shape one on slide one in the active presentation. Shape one must contain an OLE object.

```
MsgBox TypeName(ActivePresentation.Slides(1).Shapes(1).OLEFormat.Object)
```

This example displays the name of the application in which each embedded OLE object on slide one in the active presentation was created.

```
For Each s In ActivePresentation.Slides(1).Shapes
    If s.Type = msoEmbeddedOLEObject Then
        MsgBox s.OLEFormat.Object.Application.Name
    End If
Next
```

This example adds text to cell A1 on worksheet one in the Microsoft Excel workbook contained in shape three on slide one in the active presentation.

```
With ActivePresentation.Slides(1).Shapes(3)
    .OLEFormat.Object.Worksheets(1).Range("A1").Value = "New text"
End With
```

ObjectVerbs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproObjectVerbsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproObjectVerbsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproObjectVerbsA"}

Returns a **ObjectVerbs** collection that contains all the OLE verbs for the specified OLE object. Read-only.

ObjectVerbs Property Example

This example displays all the available verbs for the OLE object contained in shape one on slide two in the active presentation. For this example to work, shape one must contain an OLE object.

```
With ActivePresentation.Slides(2).Shapes(1).OLEFormat
    For Each v In .ObjectVerbs
        MsgBox v
    Next
End With
```

This example specifies that the OLE object contained in shape one on slide two in the active presentation will open when it's clicked during a slide show if "Open" is one of the OLE verbs for that object. For this example to work, shape one must contain an OLE object.

```
With ActivePresentation.Slides(2).Shapes(1)
    For Each sVerb In OLEFormat.ObjectVerbs
        If sVerb = "Open" Then
            With ActionSettings(ppMouseClick)
                .Action = ppActionOLEVerb
                .ActionVerb = sVerb
            End With
            Exit For
        End If
    Next
End With
```

ProgID Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproProgIDC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproProgIDX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproProgIDA"}

Returns the programmatic identifier (ProgID) for the specified OLE object. Read-only **String**.

programmatic identifier (ProgID)

An identifier in the form *OLEServerName*.*ObjectName* (for example, Excel.Sheet or PowerPoint.Slide) that's used by the system registry to uniquely identify an object. To specify an object created by a specific version of an application, append a period and the version number to the identifier – Excel.Sheet.8 and PowerPoint.8, for example.

ProgID Property Example

This example loops through all the objects on all the slides in the active presentation and sets all linked Microsoft Excel worksheets to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Excel.Sheet.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

BaseLineAlignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBaseLineAlignmentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBaseLineAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBaseLineAlignmentA"}

Returns or sets the base line alignment for the specified paragraph. Can be one of the following **PpBaselineAlignment** constants: **ppBaselineAlignBaseline**, **ppBaselineAlignCenter**, **ppBaselineAlignFarEast50**, **ppBaselineAlignMixed**, or **ppBaselineAlignTop**. Read/write **Long**.

BaseLineAlignment Property Example

This example displays the base line alignment for the paragraphs in shape two on slide one in the active presentation.

```
MsgBox ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .ParagraphFormat.BaseLineAlignment
```

TextDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextDirectionA"}

Middle East version only. Returns or sets the text direction for the specified paragraph. Can be one of the following **PpDirection** constants: **ppDirectionLeftToRight**, **ppDirectionMixed**, or **ppDirectionRightToLeft**. Read/write **Long**.

TextDirection Property Example

This example displays the text direction for the paragraphs in shape two on slide one in the active presentation.

```
MsgBox ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .ParagraphFormat.TextDirection
```

Alignment Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAlignmentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAlignmentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAlignmentA"}

ParagraphFormat object: Returns or sets the alignment for each paragraph in the specified paragragh format. Can be one of the following **PpParagraphAlignment** constants: **ppAlignCenter**, **ppAlignDistribute**, **ppAlignJustify**, **ppAlignLeft**, **ppAlignmentMixed**, or **ppAlignRight**. Read/write **Long**.

TextEffectFormat object: Returns or sets the alignment for the specified WordArt. Can be one of the following **MsoTextEffectAlignment** constants: **msoTextEffectAlignmentCentered**, **msoTextEffectAlignmentLeft**, **msoTextEffectAlignmentLetterJustify**, **msoTextEffectAlignmentMixed**, **msoTextEffectAlignmentRight**, **msoTextEffectAlignmentStretchJustify**, or **msoTextEffectAlignmentWordJustify**. Read/write **Long**.

Alignment Property Example

This example left aligns the paragraphs in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .ParagraphFormat.Alignment = ppAlignLeft
```

This example adds a WordArt object to slide one in the active presentation and then right aligns the WordArt.

```
Set mySh = Application.ActivePresentation.Slides(1).Shapes
Set myTE = mySh.AddTextEffect(PresetTextEffect:=msoTextEffect1, _
    Text:="Test Text", FontName:="Palatino", FontSize:=54, _
    FontBold:=True, FontItalic:=False, Left:=100, Top:=50)
myTE.TextEffect.Alignment = msoTextEffectAlignmentRight
```

Bullet Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBulletC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBulletX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBulletA"}

Returns a **BulletFormat** object that represents bullet formatting for the specified paragraph format. Read-only.

Bullet Property Example

This example sets the bullet size and bullet color for the paragraphs in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame
    With .TextRange.ParagraphFormat.Bullet
        .Visible = True
        .RelativeSize = 1.25
        .Font.Color = RGB(255, 0, 255)
    End With
End With
```

LineRuleAfter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLineRuleAfterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLineRuleAfterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLineRuleAfterA"}

True if line spacing after the last line in each paragraph is set to a specific number of lines, or **False** if line spacing is set to a specific number of points. Read/write **Long**.

LineRuleAfter Property Example

This example displays a message box that shows the setting for space after paragraphs for the text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame
    With .TextRange.ParagraphFormat
        sa = .SpaceAfter
        If .LineRuleAfter Then
            saUnits = " lines"
        Else
            saUnits = " points"
        End If
    End With
End With
MsgBox "Current spacing after paragraphs: " & sa & saUnits
```

LineRuleBefore Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLineRuleBeforeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLineRuleBeforeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLineRuleBeforeA"}

True if line spacing before the first line in each paragraph is set to a specific number of lines, or **False** if line spacing is set to a specific number of points. Read/write **Long**.

LineRuleBefore Property Example

This example displays a message box that shows the setting for space before paragraphs for text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame
    With .TextRange.ParagraphFormat
        sb = .SpaceBefore
        If .LineRuleBefore Then
            sbUnits = " lines"
        Else
            sbUnits = " points"
        End If
    End With
End With
MsgBox "Current spacing before paragraphs: " & sb & sbUnits
```

LineRuleWithin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLineRuleWithinC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLineRuleWithinX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLineRuleWithinA"}

True if line spacing between base lines is set to a specific number of lines, or **False** if line spacing is set to a specific number of points. Read/write **Long**.

LineRuleWithin Property Example

This example displays a message box that shows the setting for line spacing for text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame
    With .TextRange.ParagraphFormat
        ls = .SpaceWithin
        If .LineRuleWithin Then
            lsUnits = " lines"
        Else
            lsUnits = " points"
        End If
    End With
End With
MsgBox "Current line spacing: " & ls & lsUnits
```

SpaceWithin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSpaceWithinC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSpaceWithinX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSpaceWithinA"}

Returns or sets the amount of space between base lines in the specified text, in points or lines. Read/write **Single**.

SpaceWithin Property Example

This example sets line spacing to 21 points for the text in shape two on slide two in the active presentation.

```
With Application.ActivePresentation.Slides(2).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat
        .LineRuleWithin = False
        .SpaceWithin = 21
    End With
End With
```

SpaceAfter Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproSpaceAfterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproSpaceAfterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproSpaceAfterA"}

Returns or sets the amount of space after the last line in each paragraph of the specified text, in points or lines. Read/write **Single**.

SpaceAfter Property Example

This example sets the spacing after paragraphs to 6 points for the text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat
        .LineRuleAfter = False
        .SpaceAfter = 6
    End With
End With
```

SpaceBefore Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSpaceBeforeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSpaceBeforeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSpaceBeforeA"}

Returns or sets the amount of space before the first line in each paragraph of the specified text, in points or lines. Read/write **Single**.

SpaceBefore Property Example

This example sets the spacing before paragraphs to 6 points for the text in shape two on slide in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat
        .LineRuleBefore = False
        .SpaceBefore = 6
    End With
End With
```

FirstSlideNumber Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFirstSlideNumberC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFirstSlideNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFirstSlideNumberA"}

Returns or sets the slide number for the first slide in the presentation. Read/write **Long**.

Remarks

The slide number is the actual number that will appear in the lower-right corner of the slide when you display slide numbers. This number is determined by the number (order) of the slide within the presentation (the **SlideIndex** property value) and the starting slide number for the presentation (the **FirstSlideNumber** property value). The slide number will always be equal to the the starting slide number + the slide index number – 1. The **SlideNumber** property returns the slide number.

FirstSlideNumber Property Example

This example shows how changing the first slide number in the active presentation affects the slide number of a specific slide.

```
With Application.ActivePresentation
    .PageSetup.FirstSlideNumber = 1  'starts slide numbering at 1
    MsgBox .Slides(2).SlideNumber    'returns 2

    .PageSetup.FirstSlideNumber = 10  'starts slide numbering at 10
    MsgBox .Slides(2).SlideNumber     'returns 11
End With
```

NotesOrientation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNotesOrientationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNotesOrientationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNotesOrientationA"}

Returns or sets the on-screen and printed orientation of notes pages, handouts, and outlines for the specified presentation. Can be one of the following **MsoOrientation** constants: **msoOrientationHorizontal**, **msoOrientationMixed**, or **msoOrientationVertical**. Read/write **Long**.

NotesOrientation Property Example

This example sets the orientation of all notes pages, handouts, and outlines in the active presentation to horizontal (landscape).

```
Application.ActivePresentation.PageSetup.NotesOrientation = _
    msoOrientationHorizontal
```

SlideHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideHeightA"}

Returns or sets the slide height, in points. Read/write **Single**.

SlideHeight Property Example

This example sets the slide height to 8.5 inches and the slide width to 11 inches for the active presentation.

```
With Application.ActivePresentation.PageSetup
    .SlideWidth = 11 * 72
    .SlideHeight = 8.5 * 72
End With
```

SlideOrientation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideOrientationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideOrientationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideOrientationA"}

Returns or sets the on-screen and printed orientation of slides in the specified presentation. Can be one of the following **MsoOrientation** constants: **msoOrientationHorizontal**, **msoOrientationMixed**, or **msoOrientationVertical**. Read/write **Long**.

SlideOrientation Property Example

This example sets orientation of all slides in the active presentation to vertical (portrait).

```
Application.ActivePresentation.PageSetup.SlideOrientation = _
    msoOrientationVertical
```

SlideSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideSizeA"}

Returns or sets the slide size for the specified presentation. Can be one of the following **PpSlideSizeType** constants: **ppSlideSize35MM**, **ppSlideSizeA4Paper**, **ppSlideSizeCustom**, **ppSlideSizeLetterPaper**, **ppSlideSizeOnScreen**, or **ppSlideSizeOverhead**. Read/write **Long**.

SlideSize Property Example

This example sets the slide size to overhead for the active presentation.

```
Application.ActivePresentation.PageSetup _
    .SlideSize = ppSlideSizeOverhead
```

SlideWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideWidthA"}

Returns or sets the slide width, in points. Read/write **Single**.

SlideWidth Property Example

This example sets the slide height to 8.5 inches and the slide width to 11 inches for the active presentation.

```
With Application.ActivePresentation.PageSetup
    .SlideWidth = 11 * 72
    .SlideHeight = 8.5 * 72
End With
```

Brightness Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBrightnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBrightnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBrightnessA"}

Returns or sets the brightness of the specified picture or OLE object. The value for this property must be a number from 0.0 (dimmest) to 1.0 (brightest). Read/write **Single**.

Brightness Property Example

This example sets the brightness for shape one on `myDocument`. Shape one must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).PictureFormat.Brightness = 0.3
```

ColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproColorTypeA"}

Returns or sets the type of color transformation applied to the specified picture or OLE object. Can be one of the following **MsoPictureColorType** constants: **msoPictureAutomatic**, **msoPictureBlackAndWhite**, **msoPictureGrayscale**, **msoPictureMixed**, or **msoPictureWatermark**. Read/write **Long**.

ColorType Property Example

This example sets the color transformation to grayscale for shape one on `myDocument`. Shape one must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).PictureFormat.ColorType = msoPictureGrayScale
```

Contrast Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproContrastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproContrastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproContrastA"}

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**.

Contrast Property Example

This example sets the contrast for shape one on `myDocument`. Shape one must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).PictureFormat.Contrast = 0.8
```

IncrementBrightness Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementBrightnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementBrightnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementBrightnessA"}

Changes the brightness of the picture by the specified amount. Use the **Brightness** property to set the absolute brightness of the picture.

Syntax

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.

IncrementBrightness Method Example

This example creates a duplicate of shape one on `myDocument` and then moves and darkens the duplicate. For the example to work, shape one must be either a picture or an OLE object.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Duplicate
    .PictureFormat.IncrementBrightness -0.2
    .IncrementLeft 50
    .IncrementTop 50
End With
```

IncrementContrast Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementContrastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementContrastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementContrastA"}

Changes the contrast of the picture by the specified amount. Use the **Contrast** property to set the absolute contrast for the picture.

Syntax

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.

IncrementContrast Method Example

This example increases the contrast for all pictures on `myDocument` that aren't already set to maximum contrast.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.Type = msoPicture Then
        s.PictureFormat.IncrementContrast 0.1
    End If
Next
```

TransparencyColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTransparencyColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTransparencyColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTransparencyColorA"}

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**.

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.

TransparencyColor Property Example

This example sets the color that has the RGB value returned by the function RGB(0, 0, 255) as the transparent color for shape one on `myDocument`. For the example to work, shape one must be a bitmap.

```
blueScreen = RGB(0, 0, 255)
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1)
    With .PictureFormat
        .TransparentBackground = True
        .TransparencyColor = blueScreen
    End With
    .Fill.Visible = False
End With
```

TransparentBackground Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTransparentBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTransparentBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTransparentBackgroundA"}

True if the parts of the picture that are the color defined as the transparent color appear transparent. Use the **TransparencyColor** property to set the transparent color. Applies to bitmaps only. Read/write **Long**.

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.

TransparentBackground Property Example

This example sets the color that has the RGB value returned by the function RGB(0, 24, 240) as the transparent color for shape one on `myDocument`. For the example to work, shape one must be a bitmap.

```
blueScreen = RGB(0, 0, 255)
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1)
    With .PictureFormat
        .TransparentBackground = True
        .TransparencyColor = blueScreen
    End With
    .Fill.Visible = False
End With
```

ActionVerb Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActionVerbC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActionVerbX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActionVerbA"}

PlaySettings object: Returns or sets a string that contains the OLE verb that will be run when the specified OLE object is animated during a slide show. The default verb specifies the action that the OLE object runs – such as playing a wave file or displaying data so that the user can modify it – after the previous animation or slide transition. Read/write **String**.

ActionSetting object : Returns or sets a string that contains the OLE verb that will be run when the user clicks the specified shape or passes the mouse pointer over it during a slide show. The **Action** property must be set to **ppActionOLEVerb** first for this property to affect the slide show action. Read/write **String**.

The possible values for this property correspond to the items listed in the **Object action** box on the **Play Settings** tab in the **Custom Animation** dialog box (**Slide Show** menu) when the specified object is selected.

ActionVerb Property Example

This example specifies that shape three on slide one in the active presentation will automatically open for editing when it's animated. Shape three must be an OLE object that contains a sound or movie object and that supports the Edit verb.

```
Set OLEobj = ActivePresentation.Slides(1).Shapes(3)
With OLEobj.AnimationSettings.PlaySettings
    .PlayOnEntry = True
    .ActionVerb = "Edit"
End With
```

This example sets shape three on slide one to be played whenever the mouse pointer passes over it during a slide show.

```
With ActivePresentation.Slides(1).Shapes(3).ActionSettings(ppMouseOver)
    .ActionVerb = "Play"
    .Action = ppActionOLEVerb
End With
```

HideWhileNotPlaying Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHideWhileNotPlayingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHideWhileNotPlayingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHideWhileNotPlayingA"}

True if the specified media clip is hidden during a slide show except when it's playing. Read/write **Long**.

HideWhileNotPlaying Property Example

This example inserts a movie named "Clock.avi" onto slide one in the active presentation, sets it to play automatically after the slide transition, and specifies that the movie object be hidden during a slide show except when it's playing.

```
With ActivePresentation.Slides(1).Shapes.AddOLEObject(Left:=10, Top:=10, _
        Width:=250, Height:=250, FileName:="c:\winnt\clock.avi")
    With .AnimationSettings.PlaySettings
        .PlayOnEntry = True
        .HideWhileNotPlaying = True
    End With
End With
```

LoopUntilStopped Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLoopUntilStoppedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLoopUntilStoppedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLoopUntilStoppedA"}

PlaySettings object: **True** if the specified movie or sound loops continuously until either the next movie or sound starts, the user clicks the slide, or a slide transition occurs. Read/write **Long**.

SlideShowSettings object: **True** if the specified slide show loops continuously until the user presses ESC. Read/write **Long**.

LoopUntilStopped Property Example

This example specifies that shape three on slide one in the active presentation will start to play in the animation order and will continue to play until the next media clip starts. Shape three must be a sound or movie object.

```
ActivePresentation.Slides(1).Shapes(3).AnimationSettings.PlaySettings.LoopU
ntilStopped = True
```

This example starts a slide show of the active presentation that will automatically advance the slides (using the stored timings) and will loop continously through the presentation until the user presses ESC.

```
With ActivePresentation.SlideShowSettings
    .AdvanceMode = ppSlideShowUseSlideTimings
    .LoopUntilStopped = True
    .Run
End With
```

MediaType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro MediaTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro MediaTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro MediaTypeA"}

Returns the OLE media type. Can be one of the following **PpMediaType** constants: **ppMediaTypeMixed**, **ppMediaTypeMovie**, **ppMediaTypeOther**, or **ppMediaTypeSound**. Read-only **Long**.

MediaType Property Example

This example sets all native sound objects on slide one in the active presentation to loop until manually stopped during a slide show.

```
For Each so In ActivePresentation.Slides(1).Shapes
    If so.Type = msoMedia Then
        If so.MediaType = ppMediaTypeSound Then
            so.AnimationSettings.PlaySettings.LoopUntilStopped = True
        End If
    End If
Next
```

PauseAnimation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPauseAnimationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPauseAnimationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPauseAnimationA"}

True if the slide show pauses until the specified media clip is finished playing. **False** if the slide show continues while the media clip plays in the background. Read/write **Long**.

Remarks

For the **PauseAnimation** property setting to take effect, the **PlayOnEntry** property of the specified shape must be set to **True**.

PauseAnimation Property Example

This example specifies that shape three on slide one in the active presentation will be played automatically when it's animated and that the slide show won't continue while the movie is playing in the background. Shape three must be a sound or movie object.

```
Set OLEobj = ActivePresentation.Slides(1).Shapes(3)
With OLEobj.AnimationSettings.PlaySettings
    .PlayOnEntry = True
    .PauseAnimation = True
End With
```

PlayOnEntry Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPlayOnEntryC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPlayOnEntryX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPlayOnEntryA"}

True if the specified movie or sound is played automatically when it's animated. Read/write **Long**.

Remarks

For the **PauseAnimation** property setting to take effect, the **Animate** property of the **ActionSettings** object must be set to **True**.

Use the **ActionVerb** property to set the verb that will be invoked when the media clip is animated.

PlayOnEntry Property Example

This example specifies that shape three on slide one in the active presentation will be played automatically when it's animated. Shape three must be a sound or movie object.

```
Set OLEobj = ActivePresentation.Slides(1).Shapes(3)
OLEobj.AnimationSettings.PlaySettings.PlayOnEntry = True
```

RewindMovie Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRewindMovieC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRewindMovieX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRewindMovieA"}

True if the first frame of the specified movie is automatically redisplayed as soon as the movie has finished playing. Read/write **Long**.

RewindMovie Property Example

This example specifies that the first frame of the movie represented by shape three on slide one in the active presentation will be automatically redisplayed when the movie has finished playing. Shape three must be a movie object.

```
Set OLEobj = ActivePresentation.Slides(1).Shapes(3)
OLEobj.AnimationSettings.PlaySettings.RewindMovie = True
```

StopAfterSlides Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproStopAfterSlidesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproStopAfterSlidesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproStopAfterSlidesA"}

Returns or sets the number of slides to be displayed before the media clip stops playing. Read/write **Long**.

Remarks

For the **StopAfterSlides** property setting to take effect, the **PauseAnimation** property of the specified slide must be set to **False**, and the **PlayOnEntry** property must be set to **True**.

The media clip will stop playing when the specified number of slides have been displayed or when the clip comes to an end − whichever comes first. A value of 0 (zero) specifies that the clip will stop playing after the current slide.

StopAfterSlides Property Example

This example specifies that the media clip represented by shape three on slide one in the active presentation will be played automatically when it's animated, that the slide show will continue while the media clip is playing in the background, and that the clip will stop playing after three slides are displayed or when the end of the clip is reached – whichever comes first. Shape three must be a sound or movie object.

```
Set OLEobj = ActivePresentation.Slides(1).Shapes(3)
With OLEobj.AnimationSettings.PlaySettings
    .PlayOnEntry = True
    .PauseAnimation = False
    .StopAfterSlides = 3
End With
```

Returning an Object from a Collection

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowReturningAnObjectC"}

The **Item** method returns a single object from a collection. The following example sets the `firstPres` variable to a **Presentation** object that represents presentation one.

```
Set firstPres = Presentations.Item(1)
```

The **Item** method is the default method for most collections, so you can write the same statement more concisely by omitting the **Item** keyword.

```
Set firstPres = Presentations(1)
```

For more information about a specific collection, see the Help topic for that collection or the **Item** method for the collection.

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. Many objects are given automatically generated names when they are created. For example, the first slide you create will be automatically named "Slide1." If the first two shapes you create are a rectangle and an oval, their default names will be "Rectangle 1" and "Oval 2". You may want to give an object a more meaningful name to make it easier to refer to later. Most often, this is done by setting the object's **Name** property. The following example sets a meaningful name for a slide as it is added. The name can then be used instead of the index number to refer to the slide.

```
ActivePresentation.Slides.Add(1, 1).Name = "Home Page Slide"
With ActivePresentation.Slides("Home Page Slide")
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientDiagonalDown, 1,
msoGradientBrass
End With
```

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 a **PpTextStyleType** constant with the **Item** method of the **TextStyles** collection to return a single text style.

The following example sets the margins for the body area on slides in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).TextFrame
    .MarginBottom = 50
    .MarginLeft = 50
    .MarginRight = 50
    .MarginTop = 50
End With
```

default properties and methods

Many objects have a default property or method. Visual Basic applies the default property or method to a given object to resolve an expression that wouldn't otherwise be valid. You can write statements that use default properties or methods more concisely by omitting the default keywords. For example, the **Item** method is the default method for most collections in PowerPoint, so the following two statements are equivalent:

```
Presentations.Item(1).Slides.Item(1).Shapes.Item(1).Delete
Presentations(1).Slides(1).Shapes(1).Delete
```

Note In the Object Browser, default properties and methods are indicated by a blue dot in addition to the property or method graphic that appears before the name.

Working with shapes (drawing objects)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowWorkingWithShapesC"}

Shapes, or drawing objects, are represented by three different objects: the **Shapes** collection, the **ShapeRange** collection, and the **Shape** object. In general, you use the **Shapes** collection to create shapes and when you want to iterate through all the shapes on a slide; you use the **Shape** object when you want to modify a single shape; and you use the **ShapeRange** collection when you want to modify multiple shapes the same way you can work with multiple selected shapes in the user interface.

Setting properties for a shape

Many formatting properties of shapes aren't set by properties that apply directly to the **Shape** or **ShapeRange** object. Instead, related shape attributes are grouped under secondary objects, such as the **FillFormat** object, which contains all the properties that relate to the shape's fill, or the **LinkFormat** object, which contains all the properties that are unique to linked OLE objects. To set properties for a shape, you must first return the object that represents the set of related shape attributes and then set properties of that returned object. For example, you use the **Fill** property to return the **FillFormat** object, and then you set the **ForeColor** property of the **FillFormat** object to set the fill foreground color for the specified shape, as shown in the following example.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
```

Applying a property or method to several shapes at the same time

In the user interface, there are some operations you can perform with several shapes selected; for example, you can select several shapes and set all their individual fills at once. There are other operations you can only perform with a single shape selected; for example, you can only edit the text in a shape if a single shape is selected.

In Visual Basic, there are two ways to apply properties and methods to a set of shapes.These two ways allow you to perform any operation that you can perform on a single shape on a range of shapes, whether or not you can perform the same operation in the user interface.

- If the operation works on a multiple selected shapes in the user interface, you can perform the same operation in Visual Basic by constructing a **ShapeRange** collection that contains the shapes you want to work with, and applying the appropriate properties and methods directly to the **ShapeRange** collection.
- If the operation doesn't work on multiple selected shapes in the user interface, you can still perform the operation in Visual Basic by looping through the **Shapes** collection or through a **ShapeRange** collection that contains the shapes you want to work with, and applying the appropriate properties and methods to the individual **Shape** objects in the collection.

Many properties and methods that apply to the **Shape** object and **ShapeRange** collection fail if applied to certain kinds of shapes. For example, the **TextFrame** property fails if applied to a shape that cannot contain text. If you are not positive that each the shapes in a **ShapeRange** collection can have a certain property or method applied to it, don't apply the property or method to the **ShapeRange** collection. If you want to apply one of these properties or methods to a collection of shapes, you must loop through the collection and test each individual shape to make sure it is an appropriate type of shape before applying to property or method to it.

Applying a property or method to a ShapeRange collection

If you can perform an operation on multiple selected shapes in the user interface at the same time, you can do the programmatic equivalent by constructing a **ShapeRange** collection and then applying the appropriate properties or methods to it. The following example constructs a shape range that contains the AutoShapes named "Big Star" and "Little Star" on `myDocument` and applies a gradient

fill to them.

```
Set myDocument = ActivePresentation.Slides(1)
Set myRange = myDocument.Shapes.Range(Array("Big Star", "Little Star"))
myRange.Fill.PresetGradient msoGradientHorizontal, 1, msoGradientBrass
```

The following are general guidelines for how properties and methods behave when they're applied to a **ShapeRange** collection.

- Applying a method to a the collection is equivalent to applying the method to each individual **Shape** object in that collection.
- Setting the value of a property of the collection is equivalent to setting the value of the property of each individual shape in that range.
- A property of the collection that returns a constant returns the value of the property for an individual shape in the collection if all shapes in the collection have the same value for that property. If not all shapes in the collection have the same value for the property, it returns the "mixed" constant.
- A property of the collection that returns a simple data type (such as **Long**, **Single**, or **String**) returns the value of the property for an individual shape if all shapes in the collection have the same value for that property.
- The value of some properties can be returned or set only if there's exactly one shape in the collection. If there's more than one shape in the collection, a run-time error occurs. This is generally the case for returning or setting properties when the equivalent action in the user interface is possible only with a single shape (actions such as editing text in a shape or editing the points of a freeform).

The preceding guidelines also apply when you are setting properties of shapes that are grouped under secondary objects of the **ShapeRange** collection, such as the **FillFormat** object. If the secondary object represents operations that can be performed on multiple selected objects in the user interface, you will be able to return the object from a **ShapeRange** collection and set its properties. For example, you can use the **Fill** property to return the **FillFormat** object that represents the fills of all the shapes in the **ShapeRange** collection. Setting the properties of this **FillFormat** object will set the same properties for all the individual shapes in the **ShapeRange** collection.

Looping through a Shapes or ShapeRange collection

Even if you cannot perform an operation on several shapes in the user interface at the same time by selecting them and then using a command, you can perform the equivalent action programmatically by looping through the **Shapes** collection or through a **ShapeRange** collection that contains the shapes you want to work with, and applying the appropriate properties and methods to the individual **Shape** objects in the collection. The following example loops through all the shapes on `myDocument` and adds text to each shape that is an AutoShape.

```
Set myDocument = ActivePresentation.Slides(1)
For Each sh In myDocument.Shapes
    If sh.Type = msoAutoShape Then
            sh.TextFrame.TextRange.InsertAfter " (version 1)"
    End If
Next
```

The following example constructs **ShapeRange** collection that contains all the currently selected shapes in the active window and sets the text in each shape in the collection that can contain text.

```
For Each sh in ActiveWindow.Selection.ShapeRange
    If sh.HasTextFrame Then
        sh.TextFrame.TextRange = "Initially selected"
    End If
Next
```

Aligning, distributing, and grouping shapes in a shape range

Use the **Align** and **Distribute** methods to position a set of shapes relative to one another or relative to the document that contains them. Use the **Group** method or the **Regroup** method to form a single grouped shape from a set of shapes.

Controlling one Microsoft Office application from another

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowControllingAppsC"}

If you want to run code in one Microsoft Office application that works with the objects in another application, follow these steps.

1. Set a reference to the other application's type library in the **References** dialog box (**Tools** menu). After you have done this, the objects, properties, and methods will show up in the Object Browser and the syntax will be checked at compile time. You can also get context-sensitive Help on them.
2. Declare object variables that will refer to the objects in the other application as specific types. Make sure you qualify each type with by the name of the application that is supplying the object. For example, the following statement declares a variable that will point to a Word document and another that refers to a Microsoft Excel workbook.

   ```
   Dim appWD As Word.Application, wbXL As Excel.Workbook
   ```

   Note You must follow the steps above if you want your code to be early bound.

3. Use the **CreateObject** function with the OLE Programmatic Identifier of the object you want to work with in the other application, as shown in the following example. If you want to see the session of the other application, set the **Visible** property to **True**.

   ```
   Dim appWD As Word.Application

   Set appWD = CreateObject("Word.Application")
   appWd.Visible = True
   ```

4. Apply properties and methods to the object contained in the variable. For example, the following instruction creates a new Word document.

   ```
   Dim appWD As Word.Application

   Set appWD = CreateObject("Word.Application.8")
   appWD.Documents.Add
   ```

5. When you are done working with the other application, use the **Quit** method to close it, as shown in the following example.

   ```
   appWd.Quit
   ```

early and late binding

When you create an object variable in one application that refers to an object supplied by another application, Visual Basic must verify that the object exists and that any properties or methods used with the object are specified correctly. This verification process is known as *binding*. Binding can occur at run time (late binding) or at compile time (early binding). Late bound code is slower than early bound code. To make your code early bound, and therefore more efficient, you must set a reference to the type library that contains the objects you want to refer to, and you must declare your object variables as specific types.

Using ActiveX controls on slides

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowUsingActiveXControlsSlidesC"}

You can add controls to your slides to provide a sophisticated way to exchange information with the user while a slide show is running. For example, you could use controls on slides to allow viewers of a show designed to be run in a kiosk a way to choose options and then run a custom show based on the viewer's choices.

For general information on adding and working with controls, see Using ActiveX Controls on a Document and Creating a Custom Dialog Box.

Keep the following points in mind when you are working with controls on slides.

- A control on a slide is in design mode except when the slide show is running.
- If you want a control to appear on all slides in a presentation, add it to the slide master.
- The **Me** keyword in an event procedure for a control on a slide refers to the slide, not the control.

Writing event code for controls on slides is very similar to writing event code for controls on forms. The following procedure sets the background for the slide the button named "cmdChangeColor" is on when the button is clicked.

```
Private Sub cmdChangeColor_Click()
    With Me
        .FollowMasterBackground = Not .FollowMasterBackground
        .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientBrass
    End With
End Sub
```

You may want to use controls to provide your slide show with navigation tools that are more complex than those built into PowerPoint. For example, if you add two buttons named "cmdBack" and "cmdForward" to the slide master and write the following code behind them, all slides based on the master (and set to show master background graphics) will have these professional-looking navigation buttons that will be active during a slide show.

```
Private Sub cmdBack_Click()
    Me.Parent.SlideShowWindow.View.Previous
End Sub

Private Sub cmdForward_Click()
    Me.Parent.SlideShowWindow.View.Next
End Sub
```

To work with all the ActiveX controls on a slide without affecting the other shapes on the slide, you can construct a **ShapeRange** collection that contains only controls. You can then apply properties and methods to the entire collection or iterate through the collection to work with each control individually. The following example aligns all the controls on slide one in the active presentation and distributes them vertically.

```
With ActivePresentation.Slides(1).Shapes
    numShapes = .Count
    If numShapes > 1 Then
        numControls = 0
        ReDim ctrlArray(1 To numShapes)
        For i = 1 To numShapes
            If .Item(i).Type = msoOLEControlObject Then
                numControls = numControls + 1
                ctrlArray(numControls) = .Item(i).Name
            End If
```

```
        Next
        If numControls > 1 Then
            ReDim Preserve ctrlArray(1 To numControls)
            Set ctrlRange = .Range(ctrlArray)
            ctrlRange.Distribute msoDistributeVertically, True
            ctrlRange.Align msoAlignLefts, True
        End If
    End If
End With
```

Using ActiveX controls on a document

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowUsingControlsC"}

Just as you can add ActiveX controls to custom dialog boxes, you can add controls directly to a document when you want to provide a sophisticated way for the user to interact directly with your macro without the distraction of dialog boxes. Use the following procedure to add ActiveX controls to your document. For more specific information about using ActiveX controls in PowerPoint, see Using ActiveX controls on slides.

1. Add controls to the document

   Display the **Control Toolbox**, click the control you want to add, and then click the document.
2. Set control properties

   Right-click a control in design mode and click **Properties** to display the Properties window.
3. Initialize the controls

   You can initialize controls in a procedure.
4. 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.
5. Use control values while code is running

   Some properties can be set at run time.

Creating a custom dialog box

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowCreatingaCustomDialogBoxC"}

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. Changes made to the dialog box by the user are lost when the dialog box is closed.

Creating a UserForm

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowCreatingAUserFormC"}

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.

Adding controls to a UserForm

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowAddingControlstoaUserFormC"}

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 "look and feel" 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.

Adding controls to a document

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowAddingControlstoaDocumentC"}

To add controls to a document, display the **Control Toolbox**, click the control you want to add, and then click on the document. Drag an adjustment handle of 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 **Control Toolbox** creates a template of that control, which can be reused. This is a useful feature for implementing a standard "look and feel" for your applications.

ActiveX controls

For more information on 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
ComboBox
CommandButton
Frame
Image
Label
ListBox
MultiPage
OptionButton
ScrollBar
SpinButton
TabStrip
TextBox
ToggleButton

Setting control properties

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowSettingControlPropertiesC"}

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.

design and run modes

Microsoft Office and the Visual Basic Editor have two modes: design mode and run mode.

In design mode, you can design custom dialog boxes and controls and write code. Events do not fire and event procedures do not automatically run in design mode.

In run mode, you interact with your application the way a user would: events fire, and event procedures run. You cannot edit code in run mode.

Initializing control properties

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowInitializingControlPropertiesC"}

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 **AddItem** method to add data to a list box. 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 **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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowControlandDialogBoxEventsC"}

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 Commmand1 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 correctly before writing code.

Displaying a custom dialog box

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowDisplayingaDialogBoxC"}

To test your dialog box in the Visual Basic Editor, click **Run Sub/UserForm** on the **Run** menu in the Visual Basic Editor.

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
```

Using control values while code is running

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pphowSetControlPropertiesDynamicC"}

Some control 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"
```

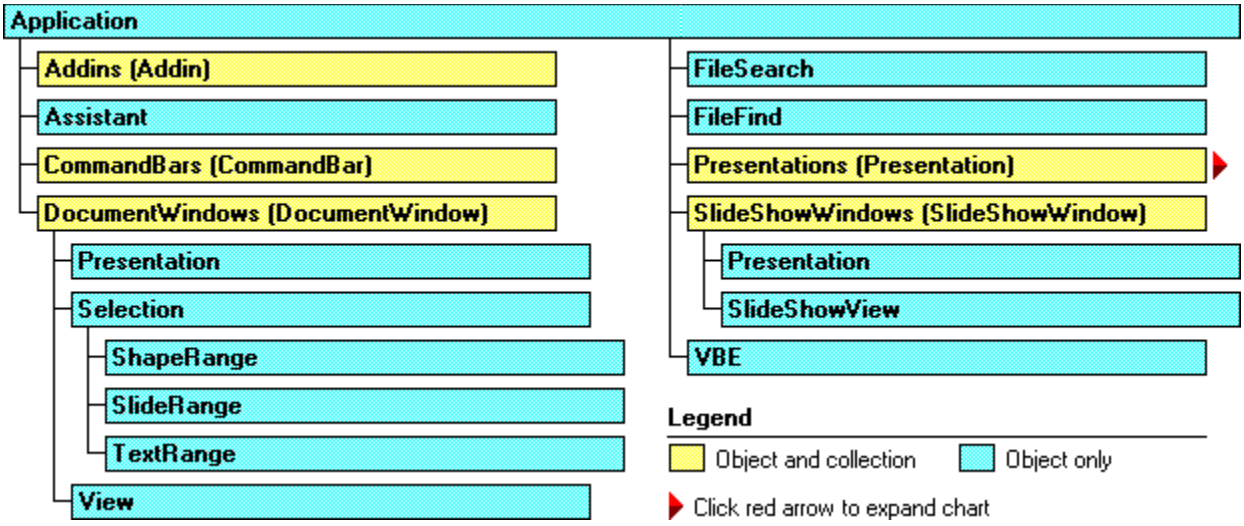
The data entered on a form by a user is lost when the form is closed. If you return the values of controls on a form after the form has been unloaded, you get the initial values for the controls rather than the values the user entered.

If you want to save the data entered on 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.

```
'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
```

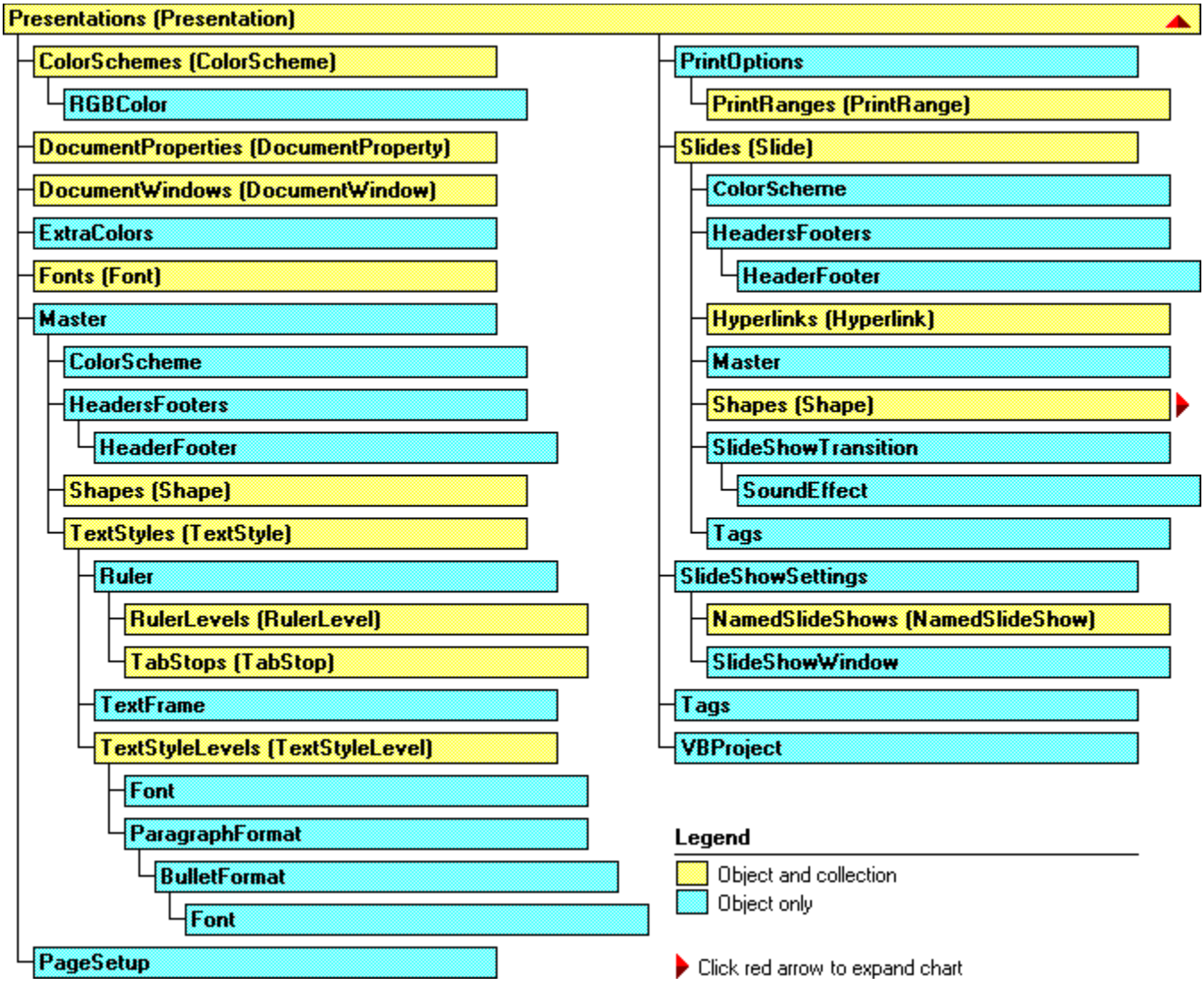
Microsoft PowerPoint Objects

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pptocObjectModelApplicationC"}



Microsoft PowerPoint Objects (Presentation)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pptocObjectModelApplicationC"}



Microsoft PowerPoint Objects (Shape)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pptocObjectModelApplicationC"}

OLE Programmatic Identifiers (Microsoft PowerPoint)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmscProgrammaticIdentifiersC;vafctCreateObject;vafctGetObject;OLE Programmatic Identifiers"}

You use an OLE Programmatic Identifier (sometimes called a ProgID) to create an Automation object. Use the OLE programmatic identifier "PowerPoint.Application.8" to create a PowerPoint 97 object. Use "PowerPoint.Application" with no version number to create an object in the most recent version of PowerPoint available on the machine where the macro is running.

Help Topic Not Available

The Help topic cannot be displayed because Visual Basic for Applications Help cannot be found or was not installed.

To install Visual Basic for Applications Help

1. Run Microsoft Office 97 Setup, and click **Add/Remove**.
2. Click **Microsoft PowerPoint**, and then click **Change Option**.
3. Click **Help**, and then click **Change Option**.
4. Make sure that the **Online Help for Visual Basic** check box is selected.
5. Continue with Setup.

Help Topic Not Available

The Help topic cannot be displayed because Microsoft Office Visual Basic Help cannot be found or was not installed.

To install Microsoft Office Visual Basic Help

1. Run Microsoft Office 97 Setup, and click **Add/Remove**.
2. Click **Microsoft PowerPoint**, and then click **Change Option**.
3. Click **Help**, and then click **Change Option**.
4. Make sure that the **Online Help for Visual Basic** check box is selected.
5. Continue with Setup.

Help Topic Not Available

The Help topic cannot be displayed because Microsoft PowerPoint Help cannot be found or was not installed.

To install Microsoft PowerPoint Help

1. Run Microsoft Office 97 Setup, and click **Add/Remove**.
2. Click **Microsoft PowerPoint**, and then click **Change Option**.
3. Click **Help**, and then click **Change Option**.
4. Make sure that the **Help for Microsoft PowerPoint** check box is selected.
5. Continue with Setup.

AddTitleMaster Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTitleMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTitleMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTitleMasterA"}

Adds a title master to the specified presentation. Returns a **Master** object that represents the title master. If the presentation already has a title master, an error occurs.

Syntax

expression.**AddTitleMaster**

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

AddTitleMaster Method Example

This example adds a title master to the active presentation if it doesn't already have one.

```
With Application.ActivePresentation
    If Not .HasTitleMaster Then .AddTitleMaster
End With
```

AddToFavorites Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddToFavoritesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddToFavoritesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddToFavoritesA"}

Presentation object: Adds a shortcut to the current selection in the specified presentation to the Favorites folder in the Windows program folder.

Hyperlink object: Adds a shortcut to the specified hyperlink's target document to the Favorites folder in the Windows program folder.

Syntax

expression.**AddToFavorites**

expression Required. An expression that returns a **Presentation** or **Hyperlink** object.

Remarks

The shortcut name is the friendly name of the document, if that's available; otherwise, the shortcut name is as calculated in HLINK.DLL.

AddToFavorites Method Example

This example adds a hyperlink to the active presentation to the Favorites folder in the Windows program folder.

```
Application.ActivePresentation.AddToFavorites
```

ApplyTemplate Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthApplyTemplateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthApplyTemplateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthApplyTemplateA"}

Applies a design template to the specified presentation.

Syntax

expression.**ApplyTemplate(*FileName*)**

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

FileName Required **String**. Specifies the name of the design template.

ApplyTemplate Method Example

This example applies the "Sparkle" design template to the active presentation.

```
Application.ActivePresentation.ApplyTemplate _
    "c:\microsoft office\templates\presentation designs\sparkle.pot"
```

BuiltInDocumentProperties Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBuiltInDocumentPropertiesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBuiltInDocumentPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBuiltInDocumentPropertiesA"}

Returns a **DocumentProperties** collection that represents all the built-in document properties for the specified presentation. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Remarks

Use the **CustomDocumentProperties** property to return the collection of custom document properties.

BuiltInDocumentProperties Property Example

This example displays the names of all the built-in document properties for the active presentation.

```
For Each p In Application.ActivePresentation.BuiltInDocumentProperties
    bidpList = bidpList & p.Name & Chr$(13)
Next
MsgBox bidpList
```

This example sets the "Category" built-in property for the the active presentation if the author of the presentation is Jake Jarmel.

```
With Application.ActivePresentation.BuiltInDocumentProperties
    If .Item("author").Value = "Jake Jarmel" Then
        .Item("category").Value = "Creative Writing"
    End If
End With
```

ColorSchemes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproColorSchemesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproColorSchemesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproColorSchemesA"}

Returns a **ColorSchemes** collection that represents the color schemes in the specified presentation. Read-only.

ColorSchemes Property Example

This example sets the background color for color scheme three in the active presentation and then applies the color scheme to all slides in the presentation that are based on the slide master.

```
With ActivePresentation
    Set cs1 = .ColorSchemes(3)
    cs1.Colors(ppBackground).RGB = RGB(128, 128, 0)
    .SlideMaster.ColorScheme = cs1
End With
```

Container Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproContainerC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproContainerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproContainerA"}

Returns the object that contains the specified embedded presentation. Read-only **Object**.

Note If the container doesn't support OLE Automation, or if the specified presentation isn't embedded, this property fails.

Container Property Example

This example hides the second section in the binder that contains the active presentation (assuming that the active presentation is embedded in an Office binder). The **Container** property of the presentation returns a **Section** object, and the **Parent** property of the **Section** object returns a **Binder** object.

```
Application.ActivePresentation.Container.Parent.Sections(2) _
    .Visible = False
```

CustomDocumentProperties Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCustomDocumentPropertiesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCustomDocumentPropertiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCustomDocumentPropertiesA"}

Returns a **DocumentProperties** collection that represents all the custom document properties for the specified presentation. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Remarks

Use the **BuiltInDocumentProperties** property to return the collection of built-in document properties.

CustomDocumentProperties Property Example

This example adds a static custom property named "Complete" for the active presentation.

```
Application.ActivePresentation.CustomDocumentProperties _
    .Add Name:="Complete", LinkToContent:=False, _
    Type:=msoPropertyTypeBoolean, Value:=False
```

This example prints out the active presentation if the value of the "Complete" custom property is **True**.

```
With Application.ActivePresentation
    If .CustomDocumentProperties("complete") Then .PrintOut
End With
```

DefaultShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDefaultShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDefaultShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDefaultShapeA"}

Returns a **Shape** object that represents the default shape for the presentation. Read-only.

DefaultShape Property Example

This example adds a shape to slide one in the active presentation, sets the default fill color to red for shapes in the presentation, and then adds another shape. This second shape will automatically have the new default fill color applied to it.

```
With Application.ActivePresentation
    Set sld1Shapes = .Slides(1).Shapes
    sld1Shapes.AddShape msoShape16pointSeal, 20, 20, 100, 100
    .DefaultShape.Fill.ForeColor.RGB = RGB(255, 0, 0)
    sld1Shapes.AddShape msoShape16pointSeal, 150, 20, 100, 100
End With
```

ExtraColors Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproExtraColorsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproExtraColorsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproExtraColorsA"}

Returns an **ExtraColors** object that represents the extra colors available in the specified presentation. Read-only.

ExtraColors Property Example

This example adds an extra color to the active presentation (if the color hasn't yet been added) and makes that color the background color for the standard color scheme.

```
With Application.ActivePresentation
    idx = .ExtraColors.Add(RGB(255, 0, 0))
    .ColorSchemes(1).Item(ppBackColor) = .ExtraColors(idx)
End With
```

FollowHyperlink Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFollowHyperlinkC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFollowHyperlinkX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFollowHyperlinkA"}

Displays a cached document, if it has already been downlaoded. Otherwise, this method resolves the hyperlink, downloads the target document and displays it in the appropriate application.

Syntax

expression.**FollowHyperlink(*Address, SubAddress, NewWindow, AddHistory, ExtraInfo, Method*)**

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

Address Required **String**. The address of the target document.

SubAddress Optional **String**. The location in the target document. By default, this argument is an empty string.

NewWindow Optional **Long**. **True** to have the target application opened in a new window. The default value is **False**.

AddHistory Optional **Long**. Not used. Reserved for future use.

ExtraInfo Optional **String**. String or byte array that specifies information for HTTP. This argument can be used, for example, to specify the coordinates of an image map or the contents of a form. It can also indicate a FAT file name. The ***Method*** argument determines how this extra information is handled.

Method Optional **Long**. Specifies the way additional information for HTTP specified by the ***ExtraInfo*** argument is handled. Can be one of the following **MsoFollowMethod** constants.

Constant	Description
msoGet	***ExtraInfo*** is a **String** that is appended to the address. This is the default value.
msoPost	***ExtraInfo*** is posted as a **String** or byte array.
msoPostFile	***ExtraInfo*** specifies a FAT file name; the file content is posted.

FollowHyperlink Method Example

This example loads the document at www.gohere.com in a new window and adds it to the histor folder.

```
Application.ActivePresentation.FollowHyperlink _
    Address:="http://www.gohere.com", NewWindow:=True, AddToHistory:=True
```

FullName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFullNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFullNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFullNameA"}

Returns the name of the specified add-in or saved presentation, including the path, the current file system separator, and the file name extension. Read-only **String**.

Remarks

This property is equivalent to the **Path** property, followed by the current file system separator, followed by the **Name** property.

FullName Property Example

This example displays the path and file name of every available add-in.

```
For Each a In Application.AddIns
   MsgBox a.FullName
Next a
```

This example displays the path and file name of the active presentation (assuming that the presentation has been saved).

```
MsgBox Application.ActivePresentation.FullName
```

HandoutMaster Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHandoutMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHandoutMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHandoutMasterA"}

Returns a **Master** object that represents the handout master. Read-only.

HandoutMaster Property Example

This example sets the background pattern on the handout master in the active presentation.

```
Application.ActivePresentation.HandoutMaster.Background.Fill _
    .Patterned msoPatternDarkHorizontal
```

HasTitleMaster Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHasTitleMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHasTitleMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHasTitleMasterA"}

True if the specified presentation has a title master. Read-only **Long**.

HasTitleMaster Property Example

This example adds a title master to the active presentation if it doesn't already have one.

```
With Application.ActivePresentation
    If Not .HasTitleMaster Then .AddTitleMaster
End With
```

LayoutDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLayoutDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLayoutDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLayoutDirectionA"}

Middle Eastern versions of PowerPoint only. Returns or sets the layout direction for the user interface. Can be one of the following **PpDirection** constants: **ppDirectionLeftToRight**, **ppDirectionMixed**, or **ppDirectionRightToLeft**. Read/write **Long**.

LayoutDirection Property Example

This example sets the layout direction to right-to-left. To use this property, you must have a Middle Eastern version of PowerPoint.

```
Application.ActivePresentation.LayoutDirection = ppDirectionRightToLeft
```

NotesMaster Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproNotesMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproNotesMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproNotesMasterA"}

Returns a **Master** object that represents the notes master. Read-only.

NotesMaster Property Example

This example sets the header and footer text for the notes master for the active presentation.

```
With Application.ActivePresentation.NotesMaster.HeadersFooters
    .Header.Text = "Employee Guidelines"
    .Footer.Text = "Volcano Coffee"
End With
```

PageSetup Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPageSetupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPageSetupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPageSetupA"}

Returns a **PageSetup** object whose properties control slide setup attributes for the specified presentation. Read-only.

PageSetup Property Example

The following example sets the slide size and slide orientation for the presentation named "Pres1."

```
With Presentations("pres1").PageSetup
    .SlideSize = ppSlideSize35MM
    .SlideOrientation = msoOrientationHorizontal
End With
```

PrintOptions Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPrintOptionsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPrintOptionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPrintOptionsA"}

Returns a **PrintOptions** object that represents print options that are saved with the specified presentation. Read-only.

PrintOptions Property Example

This example causes hidden slides in the active presentation to be printed, and it scales the printed slides to fit the paper size.

```
With Application.ActivePresentation
    With .PrintOptions
        .PrintHiddenSlides = True
        .FitToPage = True
    End With
    .PrintOut
End With
```

PrintOut Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPrintOutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPrintOutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPrintOutA"}

Prints the specified presentation.

Syntax

expression.**PrintOut(*From, To, PrintToFile, Copies, Collate*)**

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

From Optional **Long**. The number of the first page to be printed. If this argument is omitted, printing starts at the beginning of the presentation. Specifying the ***To*** and ***From*** arguments sets the contents of the **PrintRanges** object and sets the value of the **RangeType** property for the presentation.

To Optional **Long**. The number of the last page to be printed. If this argument is omitted, printing continues to the end of the presentation. Specifying the ***To*** and ***From*** arguments sets the contents of the **PrintRanges** object and sets the value of the **RangeType** property for the presentation.

PrintToFile Optional **String**. The name of the file to print to. If you specify this argument, the file is printed to a file rather than sent to a printer. If this argument is omitted, the file is sent to a printer.

Copies Optional **Long**. The number of copies to be printed. If this argument is omitted, only one copy is printed. Specifying this argument sets the value of the **NumberOfCopies** property.

Collate Optional **Long**. **True** to print a complete copy of the presentation before the first page of the next copy is printed. If this argument is omitted, multiple copies are collated. Specifying this argument sets the value of the **Collate** property.

PrintOut Method Example

This example prints two uncollated copies of each slide – whether visible or hidden – from slide two to slide five in the active presentation.

```
With Application.ActivePresentation
    .PrintOptions.PrintHiddenSlides = True
    .PrintOut From:=2, To:=5, Copies:=2, Collate:=False
End With
```

This example prints a single copy of all slides in the active presentation to the file Testprnt.prn.

```
Application.ActivePresentation.PrintOut _
    PrintToFile:="TestPrnt"
```

ReadOnly Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproReadOnlyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproReadOnlyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproReadOnlyA"}

True if the specified presentation is read-only. Read-only **Long**.

ReadOnly Property Example

If the active presentation is read-only, this example saves it as Newfile.ppt.

```
With Application.ActivePresentation
    If .ReadOnly Then .SaveAs FileName:="newfile"
End With
```

Reload Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthReloadC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthReloadX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthReloadA"}

Reloads a cached presentation by resolving the hyperlink to the workbook and downloading it. Only available with cached documents (cannot be used with FAT files).

Syntax

expression.**Reload**

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

Reload Method Example

This example reloads the presentation Testpres.ppt.

```
Presentations("testpres").Reload
```

Save Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSaveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSaveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSaveA"}

Saves the specified presentation.

Syntax

expression.**Save**

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

Remarks

Use the **SaveAs** method to save a presentation that hasn'tt been previously saved. To determine whether a presentation has been saved, test for a nonempty value for the **FullName** or **Path** property. If a document with the same name as the specified presentation already exists on disk, that document will be overwritten. No warning message will be displayed.

To mark the presentation as saved without writing it to disk, set the **Saved** property to **True**.

Save Method Example

This example saves the active presentation if it's been changed since the last time it was saved.

```
With Application.ActivePresentation
    If Not .Saved And .Path <> "" Then .Save
End With
```

SaveAs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSaveAsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSaveAsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSaveAsA"}

Saves a presentation that's never been saved, or saves a previously saved presentation under a different name.

Syntax

expression.**SaveAs(*Filename, FileFormat, EmbedFonts*)**

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

Filename Required **String**. Specifies the name to save the file under. If you don't include a full path, PowerPoint saves the file in the current folder.

FileFormat Optional **Long**. Specifies the saved file format. Can be one of the following **PpSaveAsFileType** constants: **ppSaveAsAddIn**, **ppSaveAsPowerPoint3**, **ppSaveAsPowerPoint4**, **ppSaveAsPowerPoint7**, **ppSaveAsPresentation**, **ppSaveAsRTF**, or **ppSaveAsTemplate**. If this argument is omitted, the file is saved in the format of a presentation in the current version of PowerPoint (**ppSaveAsPresentation**).

EmbedFonts Optional **Long**. **True** to have PowerPoint embed TrueType fonts in the saved presentation. The default value is **False**.

SaveAs Method Example

This example saves a copy of the active presentation under the name "New Format Copy.ppt." By default, this copy is saved in the format of a presentation in the current version of PowerPoint. The presentation is then saved as a PowerPoint 4.0 file named "Old Format Copy."

```
With Application.ActivePresentation
    .SaveCopyAs "New Format Copy"
    .SaveAs "Old Format Copy", ppSaveAsPowerPoint4
End With
```

SaveCopyAs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSaveCopyAsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSaveCopyAsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSaveCopyAsA"}

Saves a copy of the specified presentation to a file without modifying the original.

Syntax

expression.**SaveCopyAs(*FileName*)**

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

Filename Required **String**. Specifies the name to save the file under. If you don't include a full path, PowerPoint saves the file in the current folder.

SaveCopyAs Method Example

This example saves a copy of the active presentation under the name "New Format Copy.ppt." By default, this copy is saved in the format of a presentation in the current version of PowerPoint. The presentation is then saved as a PowerPoint 4.0 file named "Old Format Copy."

```
With Application.ActivePresentation
    .SaveCopyAs "New Format Copy"
    .SaveAs "Old Format Copy", ppSaveAsPowerPoint4
End With
```

Saved Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSavedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSavedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSavedA"}

True if no changes have been made to a presentation since it was last saved. Read/write **Long**.

Remarks

If the **Saved** property of a modified presentation is set to **True**, the user won't be prompted to save changes when closing the presentation, and all changes made to it since it was last saved will be lost.

Saved Property Example

This example saves the active presentation if it's been changed since the last time it was saved.

```
With Application.ActivePresentation
    If Not .Saved And .Path <> "" Then .Save
End With
```

SlideMaster Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideMasterA"}

Returns a **Master** object that represents the slide master. Read-only.

SlideMaster Property Example

This example sets the background pattern for the slide master for the active presentation.

```
Application.ActivePresentation.SlideMaster.Background.Fill _
    .PresetTextured msoTextureGreenMarble
```

Slides Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlidesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlidesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlidesA"}

Returns a **Slides** collection that represents all slides in the specified presentation. Read-only.

Slides Property Example

This example adds a slide to the active presentation.

```
Application.ActivePresentation.Slides.Add 1, ppLayoutTitle
```

SlideShowSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideShowSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideShowSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideShowSettingsA"}

Returns a **SlideShowSettings** object that represents the slide show settings for the specified presentation. Read-only.

SlideShowSettings Property Example

This example starts a slide show meant to be presented by a speaker. The slide show will run with animation and narration turned off.

```
With Application.ActivePresentation.SlideShowSettings
    .ShowType = ppShowTypeSpeaker
    .ShowWithNarration = False
    .ShowWithAnimation = False
    .Run
End With
```

TemplateName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTemplateNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTemplateNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTemplateNameA"}

Returns the name of the design template associated with the specified presentation. Read-only **String**.

Remarks

The returned string includes the MS-DOS file name extension (for file types that are registered) but doesn't include the full path.

TemplateName Property Example

The following example applies the design template Sparkle.pot to the presentation Pres1.ppt if it's not already applied to it.

```
With Presentations("pres1.ppt")
    If .TemplateName <> "sparkle.pot" Then
        .ApplyTemplate "c:\microsoft office\templates" & _
            "\presentation designs\sparkle.pot"
    End If
End With
```

TitleMaster Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTitleMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTitleMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTitleMasterA"}

Returns a **Master** object that represents the title master for the specified presentation. If the presentation doesn't have a title master, an error occurs. Read-only.

Remarks

Use the **AddTitleMaster** method to add a title master to a presentation.

TitleMaster Property Example

If the active presentation has a title master, this example sets the footer text for the title master.

```
With Application.ActivePresentation
    If .HasTitleMaster Then
        .TitleMaster.HeadersFooters.Footer.Text = "Introduction"
    End If
End With
```

Fonts Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontsA"}

Returns a **Fonts** collection that represents all fonts used in the specified presentation. Read-only.

Fonts Property Example

This example replaces the Times New Roman font with the Courier font in the active presentation.

```
Application.ActivePresentation.Fonts _
    .Replace "Times New Roman", "Courier"
```

Add Method (Presentations Collection)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPresentationsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPresentationsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPresentationsObjA"}

Creates a presentation. Returns a **Presentation** object that represents the new presentation.

Syntax

expression.**Add(*WithWindow*)**

expression Required. An expression that returns a **Presentations** collection.

WithWindow Optional **Long**. **True** to create the presentation in a visible window. If this argument is **False**, the new presentation isn't visible. The default value is **True**.

Add Method (Presentations Collection) Example

This example creates a presentation, adds a slide to it, and then saves the presentation.

```
With Presentations.Add
    .Slides.Add 1, ppLayoutTitle
    .SaveAs "Sample"
End With
```

Open Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthOpenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthOpenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthOpenA"}

Presentations collection: Opens the specified presentation. Returns a **Presentation** object that represents the opened presentation.

Syntax 1

expression.**Open(*FileName, ReadOnly, Untitled, WithWindow*)**

expression Required. An expression that returns a **Presentations** collection.

FileName Required **String**. The name of the file to open

ReadOnly Optional **Long**. **True** to open the file with read-only status. If this argument is omitted, the file is opened with read/write status.

Untitled Optional **Long**. **True** to open the file without a title. This is equivalent to creating a copy of the file. If this argument isn't specified, the file name automatically becomes the title of the opened presentation.

WithWindow Optional **Long**. **True** to open the file in a visible window. **False** to hide the opened presentation. The default value is **True**.

Open Method Example

This example opens a presentation with read-only status.

```
Presentations.Open FileName:="c:\My documents\pres1.ppt", _
    ReadOnly:=True
```

Collate Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCollateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCollateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCollateA"}

True if a complete copy of the specified presentation is printed before the first page of the next copy is printed. The default value is **True**. Read/write **Long**.

Remarks

Specifying a value for the ***Collate*** argument of the **PrintOut** method sets the value of this property.

Collate Property Example

This example prints three collated copies of the active presentation.

```
With ActivePresentation.PrintOptions
    .NumberOfCopies = 3
    .Collate = True
    .Parent.PrintOut
End With
```

FitToPage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFitToPageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFitToPageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFitToPageA"}

True if the specified slides will be scaled to fill the page they're printed on, regardless of the values in the **Height** and **Width** boxes in the **Page Setup** dialog box (**File** menu). **False** if the slides will have the dimensions specified in the **Page Setup** dialog box, whether or not those dimensions match the page they're printed on. The default value is **False**. Read/write **Long**.

FitToPage Property Example

This example prints the active presentation and scales each slide to fit the printed page.

```
With ActivePresentation
    .PrintOptions.FitToPage = True
    .PrintOut
End With
```

FrameSlides Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFrameSlidesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFrameSlidesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFrameSlidesA"}

True if a thin frame is placed around the border of the printed slides. Applies to printed slides, handouts, and notes pages. The default value is **False**. Read/write **Long**.

FrameSlides Property Example

This example prints the active presentation with a frame around each slide.

```
With ActivePresentation
    .PrintOptions.FrameSlides = True
    .PrintOut
End With
```

NumberOfCopies Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro NumberOfCopiesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro NumberOfCopiesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro NumberOfCopiesA"}

Returns or sets the number of copies of a presentation to be printed. The default value is 1. Read/write **Long**.

Remarks

Specifying a value for the ***Copies*** argument of the **PrintOut** method sets the value of this property.

NumberOfCopies Property Example

This example prints three collated copies of the active presentation.

```
With ActivePresentation.PrintOptions
    .NumberOfCopies = 3
    .Collate = True
    .Parent.PrintOut
End With
```

OutputType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOutputTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOutputTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOutputTypeA"}

Returns or sets a value that indicates which component (slides, handouts, notes pages, or an outline) of the presentation is to be printed. Can be one of the following **PpPrintOutputType** constants: **ppPrintOutputBuildSlides**, **ppPrintOutputNotesPages**, **ppPrintOutputOutline**, **ppPrintOutputSixSlideHandouts**, **ppPrintOutputSlides**, **ppPrintOutputThreeSlideHandouts**, or **ppPrintOutputTwoSlideHandouts**. The default value is **ppPrintOutputSlides**. Read/write **Long**.

OutputType Property Example

This example prints handouts of the active presentation with six slides to a page.

```
With ActivePresentation
    .PrintOptions.OutputType = ppPrintOutputSixSlideHandouts
    .PrintOut
End With
```

PrintHiddenSlides Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPrintHiddenSlidesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPrintHiddenSlidesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPrintHiddenSlidesA"}

True if the hidden slides in the specified presentation will be printed. The default value is **False**. Read/write **Long**.

PrintHiddenSlides Property Example

This example prints all slides, whether visible or hidden, in the active presentation.

```
With ActivePresentation
    .PrintOptions.PrintHiddenSlides = True
    .PrintOut
End With
```

PrintInBackground Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPrintInBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPrintInBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPrintInBackgroundA"}

True if the specified presentation is printed in the background, which means that you can continue to work while it's being printed. The default value is **True**. Read/write **Long**.

PrintInBackground Property Example

This example prints the active presentation in the background.

```
With ActivePresentation
    .PrintOptions.PrintInBackground = True
    .PrintOut
End With
```

Ranges Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRangesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRangesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRangesA"}

Returns the **PrintRanges** object, which represents the ranges of slides in the presentation to be printed. Read-only.

Remarks

If you don't want to print an entire presentation, you must use the **Add** method to create a **PrintRange** object for each consecutive run of slides you want to print. For example, if you want to print slide 1, slides 3 through 5, and slides 8 and 9 in a specified presentation, you must create three **PrintRange** objects: one that represents slide 1; one that represents slides 3 through 5; and one that represents slides 8 and 9. For more information, see the example for this property.

The **RangeType** property must be set to **ppPrintSlideRange** for the ranges in the **PrintRanges** collection to be applied.

To clear all the existing print ranges from the **PrintRanges** collection, use the **ClearAll** method.

Specifying a value for the ***To*** and ***From*** arguments of the **PrintOut** method sets the contents of the **PrintRanges** object.

Ranges Property Example

This example prints slide 1, slides 3 through 5, and slides 8 and 9 in the active presentation.

```
With ActivePresentation
    With .PrintOptions
        .RangeType = ppPrintSlideRange
        With .Ranges
            .Add 1, 1
            .Add 3, 5
            .Add 8, 9
        End With
    End With
    .PrintOut
End With
```

RangeType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRangeTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRangeTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRangeTypeA"}

PrintOptions object: Returns or sets the type of print range for the presentation. Can be one of the following **PpPrintRangeType** constants: **ppPrintAll**, **ppPrintCurrent**, **ppPrintNamedSlideShow**, **ppPrintSelection**, or **ppPrintSlideRange**. Read/write **Long**.

SlideShowSettings object: Returns or sets the type of slide show to run. Can be one of the following **PpSlideShowRangeType** constants: **ppShowAll**, **ppShowNamedSlideShow**, or **ppShowSlideRange**. Read/write **Long**.

Remarks

To print the slides ranges you've defined in the **PrintRanges** collection, you must first set the **RangeType** property to **ppPrintSlideRange**. Setting **RangeType** to anything other than **ppPrintSlideRange** means that the ranges you've defined in the **PrintRanges** collection won't be applied. However, this doesn't affect the contents of the **PrintRanges** collection in any way. That is, if you define some print ranges, set the **RangeType** property to a value other than **ppPrintSlideRange**, and then later set **RangeType** back to **ppPrintSlideRange**, the print ranges you defined before will remain unchanged.

Specifying a value for the ***To*** and ***From*** arguments of the **PrintOut** method sets the value of this property.

RangeType Property Example

This example prints the current slide the active presentation.

```
With ActivePresentation
    .PrintOptions.RangeType = ppPrintCurrent
    .PrintOut
End With
```

This example runs the named slide show "Quick Show."

```
With ActivePresentation.SlideShowSettings
    .RangeType = ppShowNamedSlideShow
    .SlideShowName = "Quick Show"
    .Run
End With
```

End Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndA"}

Returns the number of the last slide in the specified print range. Read-only **Long**.

End Property Example

This example displays a message that indicates the starting and ending slide numbers for print range one in the active presentation.

```
With ActivePresentation.PrintOptions.Ranges
    If .Count > 0 Then
        With .Item(1)
            MsgBox "Print range 1 starts on slide " & .Start & _
                " and ends on slide " & .End
        End With
    End If
End With
```

Start Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproStartC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproStartX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproStartA"}

PrintRange object: Returns the number of the first slide in the range of slides to be printed. Read-only **Long**.

TextRange object: Returns the position of the first character in the specified text range relative to the first character in the shape that contains the text. Read-only **Long**.

Start Property Example

This example displays a message that indicates the starting and ending slide numbers for print range one in the active presentation.

```
With ActivePresentation.PrintOptions.Ranges
    If .Count > 0 Then
        With .Item(1)
            MsgBox "Print range 1 starts on slide " & .Start & _
                " and ends on slide " & .End
        End With
    End If
End With
```

Add Method (PrintRanges Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPrintRangesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPrintRangesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPrintRangesObjA"}

Adds a print range to the **PrintRanges** collection. Returns a **PrintRange** object that represents a consecutive run of slides to be printed.

Syntax

expression.**Add(*Start, End*)**

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

Start Required **Long**. The first slide in the range of slides to be printed. Must be less than or equal to the value of the ***End*** argument.

End Required **Long**. The last slide in the range of slides to be printed. Must be less than or equal to the value of the ***Start*** argument.

Remarks

The **RangeType** property of the **PrintOptions** object must be set to **ppPrintSlideRange** for the ranges in the **PrintRanges** collection to be applied.

If you don't want to print an entire presentation, add print ranges to specify which slides you want to print. You must add one print range for each consecutive run of slides to be printed. For example, if you want to print slide 1, slides 3 through 5, and slides 8 and 9, you must add three print range objects. For more information, see the example for this method.

Use the **ClearAll** method to clear previously defined print ranges.

Add Method (PrintRanges Collection Object) Example

This example clears any previously defined print ranges and then prints slide 1, slides 3 through 5, and slides 8 and 9 in the active presentation.

```
With ActivePresentation.PrintOptions
    .RangeType = ppPrintSlideRange
    With .Ranges
        .ClearAll
        .Add 1, 1
        .Add 3, 5
        .Add 8, 9
    End With
End With
ActivePresentation.PrintOut
```

ClearAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthClearAllC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthClearAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthClearAllA"}

Clears all the print ranges from the **PrintRanges** collection. Use the **Add** method of the **PrintRanges** collection to add print ranges to the collection.

Syntax

expression.**ClearAll**

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

ClearAll Method Example

This example clears any previously defined print ranges in the active presentation; creates new print ranges that contain slide 1, slides 3 through 5, and slides 8 and 9; prints the newly defined slide ranges; and then clears the new print ranges.

```
With ActivePresentation.PrintOptions
    .RangeType = ppPrintSlideRange
    With .Ranges
        .ClearAll
        .Add 1, 1
        .Add 3, 5
        .Add 8, 9
        .Parent.Parent.PrintOut
        .ClearAll
    End With
End With
```

Levels Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLevelsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLevelsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLevelsA"}

Ruler object: Returns a **RulerLevels** object that represents outline indent formatting. Read-only.

TextStyle object: Returns a **TextStyleLevels** object that represents outline text formatting. Read-only.

Levels Property Example

This example sets the first-line indent and hanging indent for outline level one in body text on the slide master for the active presentation, and then it sets the the font name and font size for text at that level.

```
With Application.ActivePresentation.SlideMaster.TextStyles(ppBodyStyle)
    With .Ruler.Levels(1) ' sets indents for level 1
        .FirstMargin = 9
        .LeftMargin = 54
    End With
    With .Levels(1).Font  ' sets text formatting for level 1
        .Name = "arial"
        .Size = 36
    End With
End With
```

TabStops Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTabStopsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTabStopsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTabStopsA"}

Returns a **TabStops** collection that represents the tab stops for the specified text. Read-only.

TabStops Property Example

This example adds a slide with two text columns to the active presentation, sets a left-aligned tab stop for the title on the new slide, and assigns text that includes a tab character to the title.

```
With Application.ActivePresentation.Slides _
    .Add(2, ppLayoutTwoColumnText).Shapes
    With .Title.TextFrame
        With .Ruler
            .Levels(1).FirstMargin = 0
            .TabStops.Add ppTabStopLeft, 50
        End With
        .TextRange = "first col" + Chr(9) + "second col"
    End With
End With
```

FirstMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFirstMarginC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFirstMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFirstMarginA"}

Returns or sets the first-line indent for the specified outline level, in points. Read/write **Single**.

Remarks

If a paragraph begins with a bullet, the bullet position is determined by the **FirstMargin** property, and the position of the first text character in the paragraph is determined by the **LeftMargin** property.

FirstMargin Property Example

This example sets the first-line indent and hanging indent for outline level one in body text on the slide master for the active presentation.

```
With Application.ActivePresentation.SlideMaster.TextStyles(ppBodyStyle)
    With .Ruler.Levels(1)
        .FirstMargin = 9
        .LeftMargin = 54
    End With
End With
```

LeftMargin Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLeftMarginC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLeftMarginX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLeftMarginA"}

Returns or sets the left indent for the specified outline level, in points. Read/write **Single**.

Remarks

If a paragraph begins with a bullet, the bullet position is determined by the **FirstMargin** property, and the position of the first text character in the paragraph is determined by the **LeftMargin** property.

LeftMargin Property Example

This example sets the first-line indent and hanging indent for outline level one in body text on the slide master for the active presentation.

```
With Application.ActivePresentation.SlideMaster.TextStyles(ppBodyStyle)
    With .Ruler.Levels(1)
        .FirstMargin = 9
        .LeftMargin = 54
    End With
End With
```

ShapeRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShapeRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShapeRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShapeRangeA"}

Returns a **ShapeRange** object that represents all the slide objects that have been selected on the specified slide. This range can contain the drawings, shapes, OLE objects, pictures, text objects, titles, headers, footers, slide number placeholder, and date and time objects on a slide. Read-only.

Remarks

You can return a shape range from a selection only when the presentation is in slide view.

ShapeRange Property Example

This example sets the fill foreground color for all the selected shapes in window one.

```
Windows(1).Selection.ShapeRange.Fill.ForeColor.RGB = RGB(255, 0, 255)
```

SlideRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideRangeA"}

Returns a **SlideRange** object that represents a range of selected slides. Read-only.

Remarks

A slide range can be constructed in slide view, slide sorter view, or outline view. In slide view, **SlideRange** returns one slide – the current, displayed slide.

SlideRange Property Example

This example sets the background scheme color for all the selected slides in window one.

```
Windows(1).Selection.SlideRange.ColorScheme.Colors(ppBackground).RGB =
RGB(0, 0, 255)
```

TextRange Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextRangeA"}

Selection object: Returns a **TextRange** object that represents the selected text. Read-only.

TextFrame object: Returns a **TextRange** object that represents the text in the specified text frame. Read-only.

Remarks

You can construct a text range from a selection only when the presentation is in slide view or outline view.

TextRange Property Example

This example makes the selected text in window one bold.

```
Windows(1).Selection.TextRange.Font.Bold = True
```

Unselect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUnselectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUnselectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUnselectA"}

Cancels the current selection.

Syntax

expression.**Unselect**

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

Unselect Method Example

This example cancels the current selection in window one.

```
Windows(1).Selection.Unselect
```

IncrementOffsetX Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementOffsetXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementOffsetXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementOffsetXA"}

Changes the horizontal offset of the shadow by the specified number of points. Use the **OffsetX** property to set the absolute horizontal shadow offset.

Syntax

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.

IncrementOffsetX Method Example

This example moves the shadow on shape three on `myDocument` to the left by 3 points.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(3).Shadow.IncrementOffsetX -3
```

IncrementOffsetY Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementOffsetYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementOffsetYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementOffsetYA"}

Changes the vertical offset of the shadow by the specified number of points. Use the **OffsetY** property to set the absolute vertical shadow offset.

Syntax

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.

IncrementOffsetY Method Example

This example moves the shadow on shape three on `myDocument` up by 3 points.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(3).Shadow.IncrementOffsetY -3
```

Obscured Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproObscuredC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproObscuredX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproObscuredA"}

True if the shadow of the specified shape appears filled in and is obscured by the shape, even if the shape has no fill. **False** 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 **Long**.

Obscured Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
    .Obscured = True
End With
```

OffsetX Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOffsetXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOffsetXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOffsetXA"}

Returns or sets the horizontal offset of the shadow from the specified shape, in points. A positive value offsets the shadow to the right of the shape; a negative value offsets it to the left. Read/write **Single**.

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** method or the **IncrementOffsetY** method.

OffsetX Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```

OffsetY Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOffsetYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOffsetYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOffsetYA"}

Returns or sets the vertical offset of the shadow from the specified shape, in points. A positive value offsets the shadow to the right of the shape; a negative value offsets it to the left. Read/write **Single**.

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** method or the **IncrementOffsetY** method.

OffsetY Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .OffsetX = 5
    .OffsetY = -3
End With
```

Transparency Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTransparencyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTransparencyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTransparencyA"}

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**.

Remarks

The value of this property affects the appearance of solid-colored fills and lines only; it has no effect on the appearance of patterned lines or patterned, gradient, picture, or textured fills.

Transparency Property Example

This example sets the shadow of shape three on `myDocument` to semitransparent red. If the shape doesn't already have a shadow, this example adds one to it.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Shadow
    .Visible = True
    .ForeColor.RGB = RGB(255, 0, 0)
    .Transparency = 0.5
End With
```

ActionSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproActionSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproActionSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproActionSettingsA"}

Returns an **ActionSettings** object that contains information about what action occurs when the user clicks or moves the mouse over the specified shape or text range during a slide show. Read-only.

ActionSettings Property Example

The following example sets the actions for clicking and moving the mouse over shape one on slide two in the active presentation.

```
Set myShape = ActivePresentation.Slides(2).Shapes(1)
myShape.ActionSettings(ppMouseClick).Action = ppActionLastSlide
myShape.ActionSettings(ppMouseOver).SoundEffect.Name = "applause.wav"
```

Adjustments Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAdjustmentsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAdjustmentsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAdjustmentsA"}

Returns an **Adjustments** object that contains adjustment values for all the adjustments in the specified shape. Applies to any **Shape** or **ShapeRange** object that represents an AutoShape, WordArt, or a connector. Read-only.

Adjustments Property Example

This example sets to 0.25 the value of adjustment one on shape three on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(3).Adjustments(1) = 0.25
```

Align Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAlignC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAlignX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAlignA"}

Aligns the shapes in the specified range of shapes.

Syntax

expression.**Align(*AlignCmd, RelativeTo*)**

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

AlignCmd Required **Long**. Specifies the way the shapes in the specified shape range are to be aligned. Can be one of the following **MsoAlignCmd** constants: **msoAlignBottoms**, **msoAlignCenters**, **msoAlignLefts**, **msoAlignMiddles**, **msoAlignRights**, or **msoAlignTops**.

RelativeTo Required **Long**. **True** to align shapes relative to the edge of the document. **False** to align shapes relative to one another.

Align Method Example

This example aligns the left edges of all the shapes in the specified range in `myDocument` with the left edge of the leftmost shape in the range.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Range.Align msoAlignLefts, False
```

AnimationSettings Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAnimationSettingsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAnimationSettingsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAnimationSettingsA"}

Returns an **AnimationSettings** object that represents all the special effects you can apply to the animation of the specified shape. Read-only.

AnimationSettings Property Example

This example sets shape one on slide two in the active presentation to fly in from the left when the slide is built.

```
With ActivePresentation.Slides(2).Shapes(1).AnimationSettings
    .EntryEffect = ppEffectFlyFromLeft
    .TextLevelEffect = ppAnimateByAllLevels
End With
```

Apply Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthApplyC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthApplyX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthApplyA"}

Applies to the specified shape formatting that's been copied by using the **PickUp** method.

Syntax

expression.**Apply**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Apply Method Example

This example copies the formatting of shape one on `myDocument` and then applies the copied formatting to shape two.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```

AutoShapeType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoShapeTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoShapeTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoShapeTypeA"}

Returns or sets the shape type for the specified **Shape** or **ShapeRange** object, which must represent an AutoShape other than a line, freeform drawing, or connector. Read/write **Long**.

Note When you change the type of a shape, the shape retains its size, color, and other attributes.

Can be one of the following **MsoAutoShapeType** constants:

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
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

msoShapeDonut
msoShapeDoubleBrace
msoShapeDoubleBracket
msoShapeDoubleWave
msoShapeDownArrow
msoShapeDownArrowCallout
msoShapeDownRibbon
msoShapeExplosion1
msoShapeExplosion2
msoShapeFlowchartAlternateProcess
msoShapeFlowchartCard
msoShapeFlowchartCollate
msoShapeFlowchartConnector
msoShapeFlowchartData
msoShapeFlowchartDecision
msoShapeFlowchartDelay
msoShapeFlowchartDirectAccessStorage
msoShapeFlowchartDisplay
msoShapeFlowchartDocument
msoShapeFlowchartExtract
msoShapeFlowchartInternalStorage
msoShapeFlowchartMagneticDisk
msoShapeFlowchartManualInput
msoShapeFlowchartManualOperation
msoShapeFlowchartMerge
msoShapeFlowchartMultidocument
msoShapeFlowchartOffpageConnector
msoShapeFlowchartOr
msoShapeFlowchartPredefinedProcess
msoShapeFlowchartPreparation
msoShapeFlowchartProcess
msoShapeFlowchartPunchedTape
msoShapeFlowchartSequentialAccessStorage
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

Remarks

Use the **Type** property of the **ConnectorFormat** object to set or return the connector type.

AutoShapeType Property Example

This example replaces all 16-point stars with 32-point stars in `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.AutoShapeType = msoShape16pointStar Then
        s.AutoShapeType = msoShape32pointStar
    End If
Next
```

BlackWhiteMode Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBlackWhiteModeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBlackWhiteModeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBlackWhiteModeA"}

Returns or sets a value that indicates how the specified shape appears when the document is viewed in black-and-white mode. Can be one of the following **MsoBlackWhiteMode** constants: **msoBlackWhiteAutomatic**, **msoBlackWhiteBlack**, **msoBlackWhiteBlackTextAndLine**, **msoBlackWhiteDontShow**, **msoBlackWhiteGrayOutline**, **msoBlackWhiteGrayScale**, **msoBlackWhiteHighContrast**, **msoBlackWhiteInverseGrayScale**, **msoBlackWhiteLightGrayScale**, **msoBlackWhiteMixed**, or **msoBlackWhiteWhite**. Read/write **Long**.

BlackWhiteMode Property Example

This example sets shape one on `myDocument` to appear in black-and-white mode. When you view the document in black-and-white mode, shape one will appear black, regardless of what color it is in color mode.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).BlackWhiteMode = msoBlackWhiteBlack
```

Callout Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproCalloutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproCalloutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproCalloutA"}

Returns a **CalloutFormat** object that contains callout formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent line callouts. Read-only.

Callout Property 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 = ActivePresentation.Slides(1)
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
```

ConnectionSiteCount Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproConnectionSiteCountC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproConnectionSiteCountX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproConnectionSiteCountA"}

Returns the number of connection sites on the specified shape. Read-only **Long**.

ConnectionSiteCount Property Example

This example adds two rectangles to `myDocument` and joins them with two connectors. The beginnings of both connectors attach to connection site one on the first rectangle; the ends of the connectors attach to the first and last connection sites of the second rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
lastsite = secondRect.ConnectionSiteCount
With s.AddConnector(msoConnectorCurve, 0, 0, 100, 100).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
End With
With s.AddConnector(msoConnectorCurve, 0, 0, 100, 100).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=lastsite
End With
```

Connector Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproConnectorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproConnectorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproConnectorA"}

True if the specified shape is a connector. Read-only **Long**.

Connector Property Example

This example deletes all connectors on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    For i = .Count To 1 Step -1
        With .Item(i)
            If .Connector Then .Delete
        End With
    Next
End With
```

ConnectorFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproConnectorFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproConnectorFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproConnectorFormatA"}

Returns a **ConnectorFormat** object that contains connector formatting properties. Applies to **Shape** or **ShapeRange** objects that represent connectors. Read-only.

ConnectorFormat Property Example

This example adds two rectangles to `myDocument`, attaches them with a connector, automatically reroutes the connector along the shortest path, and then detaches the connector from the rectangles.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 0, 0).ConnectorFormat
    .BeginConnect firstRect, 1
    .EndConnect secondRect, 1
    .Parent.RerouteConnections
    .BeginDisconnect
    .EndDisconnect
End With
```

Fill Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFillC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFillX":1}
{ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFillA"}

Returns a **FillFormat** object that contains fill formatting properties for the specified shape. Read-only.

Fill Property 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 = ActivePresentation.Slides(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
```

Flip Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFlipA"}

Flips the specified shape around it's horizontal or vertical axis.

Syntax

expression.**Flip(*FlipCmd*)**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

FlipCmd Required **Long**. Specifies whether the shape is to be flipped horizontally or vertically. Can be either of the following **MsoFlipCmd** constants: **msoFlipHorizontal** or **msoFlipVertical**.

Flip Method Example

This example adds a triangle to `myDocument`, duplicates the triangle, and then flips the duplicate triangle vertically and makes it red.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeRightTriangle, 10, 10, 50,
50).Duplicate
    .Fill.ForeColor.RGB = RGB(255, 0, 0)
    .Flip msoFlipVertical
End With
```

Group Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthGroupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthGroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthGroupA"}

Groups the shapes in the specified range. Returns the grouped shapes as a single **Shape** object.

Syntax

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.

Group Method 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 = ActivePresentation.Slides(1)
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
```

GroupItems Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproGroupItemsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproGroupItemsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproGroupItemsA"}

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.

GroupItems Property 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 = ActivePresentation.Slides(1)
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
```

HorizontalFlip Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHorizontalFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHorizontalFlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHorizontalFlipA"}

True if the specified shape is flipped around the horizontal axis. Read-only **Long**.

HorizontalFlip Property Example

This example restores each shape on `myDocument` to its original state if it's been flipped horizontally or vertically.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.HorizontalFlip Then s.Flip msoFlipHorizontal
    If s.VerticalFlip Then s.Flip msoFlipVertical
Next
```

IncrementLeft Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementLeftA"}

Moves the specified shape horizontally by the specified number of points.

Syntax

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.

IncrementLeft 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

IncrementRotation Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementRotationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementRotationXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementRotationA"}

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.

Syntax

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.

IncrementRotation 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

IncrementTop Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementTopA"}

Moves the specified shape vertically by the specified number of points.

Syntax

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.

IncrementTop 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).Duplicate
    .Fill.PresetTextured msoTextureGranite
    .IncrementLeft 70
    .IncrementTop -50
    .IncrementRotation 30
End With
```

Item Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproItemC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproItemX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproItemA"}

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**.

Syntax

expression.**Item(*Index*)**

expression Required. An expression that returns an **Adjustments** object.

Index Required **Long**. The index number of the adjustment.

Remarks

AutoShapes, connectors, and WordArt objects can have up to eight adjustments.

Item Property Example

This example adds two crosses to `myDocument` and then sets the value for adjustment one (the only one on this type of AutoShape) on each cross.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    .AddShape(msoShapeCross, 10, 10, 100, 100).Adjustments(1) = 0.4
    .AddShape(msoShapeCross, 150, 10, 100, 100).Adjustments(1) = 0.2
End With
```

Line Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLineA"}

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.

Line Property Example

This example adds a blue dashed line to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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 = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeCross, 10, 10, 50, 70).Line
    .Weight = 8
    .ForeColor.RGB = RGB(255, 0, 0)
End With
```

LinkFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLinkFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLinkFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLinkFormatA"}

Returns a **LinkFormat** object that contains the properties that are unique to linked OLE objects. Read-only.

LinkFormat Property Example

This example updates and then breaks the links between any OLE objects on slide one in the active presentation and their source files.

```
For Each sh In ActivePresentation.Slides(1).Shapes
    If sh.Type = msoLinkedOLEObject Then
        With sh.LinkFormat
            .Update
            .BreakLink
        End With
    End If
Next
```

LockAspectRatio Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLockAspectRatioC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLockAspectRatioX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLockAspectRatioA"}

True if the specified shape retains its original proportions when you resize it. **False** if you can change the height and width of the shape independently of one another when you resize it. Read/write **Long**.

LockAspectRatio Property Example

This example adds a cube to `myDocument`. The cube can be moved and resized, but not reproportioned.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape(msoShapeCube, 50, 50, 100, 200).LockAspectRatio
= True
```

Nodes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNodesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNodesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNodesA"}

Returns a **ShapeNodes** collection that represents the geometric description of the specified shape. Applies to **Shape** or **ShapeRange** objects that represent freeform drawings.

Nodes Property Example

This example adds a smooth node with a curved segment after node four in shape three on `myDocument`. Shape three must be a freeform drawing with at least four nodes.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Nodes
    .Insert 4, msoSegmentCurve, msoEditingSmooth, 210, 100
End With
```

OLEFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOLEFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOLEFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOLEFormatA"}

Returns an **OLEFormat** object that contains OLE formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent OLE objects. Read-only

OLEFormat Property Example

This example loops through all the objects on all the slides in the active presentation and sets all linked Microsoft Word 97 documents to be updated manually.

```
For Each sld In ActivePresentation.Slides
    For Each sh In sld.Shapes
        If sh.Type = msoLinkedOLEObject Then
            If sh.OLEFormat.ProgID = "Word.Document.8" Then
                sh.LinkFormat.AutoUpdate = ppUpdateOptionManual
            End If
        End If
    Next
Next
```

PickUp Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPickUpC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPickUpX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPickUpA"}

Copies the formatting of the specified shape. Use the **Apply** method to apply the copied formatting to another shape.

Syntax

expression.**PickUp**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

PickUp Method Example

This example copies the formatting of shape one on `myDocument` and then applies the copied formatting to shape two.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument
    .Shapes(1).PickUp
    .Shapes(2).Apply
End With
```

PictureFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPictureFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPictureFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPictureFormatA"}

Returns a **PictureFormat** object that contains picture formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent pictures or OLE objects. Read-only.

PictureFormat Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).PictureFormat
    .Brightness = 0.3
    .Contrast = .75
End With
```

RerouteConnections Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRerouteConnectionsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRerouteConnectionsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRerouteConnectionsA"}

Reroutes connectors so that they take the shortest possible path between the shapes they connect. To do this, the **RerouteConnections** method may detach the ends of a connector and reattach them to different connecting sites on the connected shapes.

This method reroutes all connectors attached to the specified shape; if the specified shape is a connector, it's rerouted.

Syntax

expression.**RerouteConnections**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Remarks

If this method is applied to a connector, only that connector will be rerouted. If this method is applied to a connected shape, all connectors to that shape will be rerouted.

RerouteConnections Method Example

This example adds two rectangles to `myDocument`, connects them with a curved connector, and then reroutes the connector so that it takes the shortest possible path between the two rectangles. Note that the **RerouteConnections** method adjusts the size and position of the connector and determines which connecting sites it attaches to, so the values you initially specify for the ***ConnectionSite*** arguments used with the **BeginConnect** and **EndConnect** methods are irrelevant.

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
Set newConnector = s.AddConnector(msoConnectorCurve, 0, 0, 100, 100)
With newConnector.ConnectorFormat
    .BeginConnect firstRect, 1
    .EndConnect secondRect, 1
End With
newConnector.RerouteConnections
```

Rotation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproRotationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproRotationXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproRotationA"}

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.

Rotation Property Example

This example matches the rotation of all shapes on `myDocument` to the rotation of shape one.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    sh1Rotation = .Item(1).Rotation
    For o = 1 To .Count
        .Item(o).Rotation = sh1Rotation
    Next
End With
```

ScaleHeight Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthScaleHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthScaleHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthScaleHeightA"}

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.

Syntax

expression.**ScaleHeight(*Factor, RelativeToOriginalSize, fScale*)**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

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 **Long**. **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.

fScale Optional **Long**. The part of the shape retains its position when the shape is scaled. Can be one of the following **MsoScaleFrom** constants: **msoScaleFromBottomRight**, **msoScaleFromMiddle**, or **msoScaleFromTopLeft**. The default value is **msoScaleFromTopLeft**.

ScaleHeight Method 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 = ActivePresentation.Slides(1)
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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthScaleWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthScaleWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthScaleWidthA"}

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.

Syntax

expression.**ScaleWidth(*Factor, RelativeToOriginalSize, fScale*)**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

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 **Long**. **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.

fScale Optional **Long**. The part of the shape retains its position when the shape is scaled. Can be one of the following **MsoScaleFrom** constants: **msoScaleFromBottomRight**, **msoScaleFromMiddle**, or **msoScaleFromTopLeft**. The default value is **msoScaleFromTopLeft**.

ScaleWidth Method 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 = ActivePresentation.Slides(1)
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
```

TextEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextEffectA"}

Returns a **TextEffectFormat** object that contains text-effect formatting properties for the specified shape. Applies to **Shape** or **ShapeRange** objects that represent WordArt. Read-only.

TextEffect Property Example

This example sets the font style to bold for shape three on `myDocument` if the shape is WordArt.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontBold = True
    End If
End With
```

TextFrame Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextFrameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextFrameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextFrameA"}

Returns a **TextFrame** object that contains the alignment and anchoring properties for the specified shape or master text style.

Remarks

Use the **TextRange** property of the **TextFrame** object to return the text in the text frame.

Use the **HasTextFrame** property to determine whether a shape contains a text frame before you apply the **TextFrame** property.

TextFrame Property Example

This example adds a rectangle to `myDocument`, adds text to the rectangle, and sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

ThreeD Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproThreeDC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproThreeDX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproThreeDA"}

Returns a **ThreeDFormat** object that contains 3-D – effect formatting properties for the specified shape. Read-only.

ThreeD Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)    ' RGB value for purple
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```

Ungroup Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthUngroupC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthUngroupX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthUngroupA"}

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.

Syntax

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.

Ungroup Method Example

This example ungroups any grouped shapes and disassembles any pictures or OLE objects on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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 = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.Type = msoGroup Then s.Ungroup
Next
```

VerticalFlip Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVerticalFlipC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVerticalFlipX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVerticalFlipA"}

True if the specified shape is flipped around the vertical axis. Read-only **Long**.

VerticalFlip Property Example

This example restores each shape on `myDocument` to its original state if it's been flipped horizontally or vertically.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.HorizontalFlip Then s.Flip msoFlipHorizontal
    If s.VerticalFlip Then s.Flip msoFlipVertical
Next
```

Vertices Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVerticesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVerticesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVerticesA"}

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 to the **AddCurve** method or **AddPolyline** method. Read-only **Variant**.

The following table shows how the **Vertices** property associates the values in the array `vertArray()` with the coordinates of a triangle's vertices.

vertArray element	Contains
`vertArray(1, 1)`	The horizontal distance from the first vertex to the left side of the document
`vertArray(1, 2)`	The vertical distance from the first vertex to the top of the document
`vertArray(2, 1)`	The horizontal distance from the second vertex to the left side of the document
`vertArray(2, 2)`	The vertical distance from the second vertex to the top of the document
`vertArray(3, 1)`	The horizontal distance from the third vertex to the left side of the document
`vertArray(3, 2)`	The vertical distance from the third vertex to the top of the document

Vertices Property Example

This example assigns the vertex coordinates for shape one on `myDocument` to the array variable `vertArray()` and displays the coordinates for the first vertex.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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 on `myDocument`. Shape one must contain 3*n*+1 vertices for this example to succeed.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes
    .AddCurve .Item(1).Vertices
End With
```

ZOrder Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthZOrderC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthZOrderX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthZOrderA"}

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).

Syntax

expression.**ZOrder(*ZOrderCmd*)**

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

ZOrderCmd Required **Long**. Specifies where to move the specified shape relative to the other shapes. Can be one of the following **MsoZOrderCmd** constants: **msoBringForward**, **msoBringToFront**, **msoSendBackward**, or **msoSendToBack**. The constants **msoBringInFrontOfText** and **msoSendBehindText** are for use in Microsoft Word only.

Remarks

Use the **ZOrderPosition** property to determine a shape's current position in the z-order.

ZOrder Method 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 = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeOval, 100, 100, 100, 300)
    While .ZOrderPosition > 2
        .ZOrder msoSendBackward
    Wend
End With
```

ZOrderPosition Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproZOrderPositionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproZOrderPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproZOrderPositionA"}

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.

ZOrderPosition Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeOval, 100, 100, 100, 300)
    While .ZOrderPosition > 2
        .ZOrder msoSendBackward
    Wend
End With
```

AddCallout Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddCalloutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddCalloutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddCalloutA"}

Creates a borderless line callout. Returns a **Shape** object that represents the new callout.

Syntax

expression.**AddCallout(*Type, Left, Top, Width, Height*)**

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

Type Required **Long**. The type of callout line. Can be one of the following **MsoCalloutType** constants: **msoCalloutOne** (a single-segment callout line that can be either horizontal or vertical), **msoCalloutTwo** (a single-segment callout line that rotates freely), **msoCalloutThree** (a two-segment line), or **msoCalloutFour** (a three-segment line).

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the callout's bounding box relative to the upper-left corner of the document.

Width, ***Height*** Required **Single**. The width and height of the callout's bounding box, in points.

Remarks

You can insert a greater variety of callouts by using the **AddShape** method.

AddCallout Method Example

This example adds a borderless callout with a freely rotating one-segment callout line to `myDocument` and then sets the callout angle to 30 degrees.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddCallout(msoCalloutTwo, 50, 50, 200, 100).Callout.Angle
= msoCalloutAngle30
```

AddComment Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddCommentC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddCommentX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddCommentA"}

Adds a comment. Returns a **Shape** object that represents the new comment.

Syntax

expression.**AddComment(*Left, Top, Width, Height*)**

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

Left, ***Top*** Optional **Single**. The position (in points) of the upper-left corner of the comment bounding box relative to the upper-left corner of the document. By default, the comment is placed in the upper-left corner of the document.

Width, ***Height*** Optional **Single**. The width and height of the comment, in points. By default, the comment is 100 points high and 100 points wide.

AddComment Method Example

This example adds a comment that contains the text "Test Comment" to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddComment(100, 100, 150, 150)
    .TextFrame.TextRange.Text = .TextFrame.TextRange.Text + "Test Comment"
End With
```

AddConnector Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddConnectorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddConnectorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddConnectorA"}

Creates a connector. Returns a **Shape** object that represents the new connector. When a connector is added, it's not connected to anything. Use the **BeginConnect** and **EndConnect** methods to attach the beginning and end of a connector to other shapes in the document.

Syntax

expression.**AddConnector(*Type, BeginX, BeginY, EndX, EndY*)**

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

Type Required **Long**. The type of connector. Can be one of the following **MsoConnectorType** constants: **msoConnectorCurve**, **msoConnectorElbow**, or **msoConnectorStraight**.

BeginX, ***BeginY*** Required **Single**. The position (in points) of the connector's starting point relative to the upper-left corner of the document.

EndX, ***EndY*** Required **Single**. The position (in points) of the connector's end point relative to the upper-left corner of the document.

Remarks

When you attach a connector to a shape, the size and position of the connector are automatically adjusted, if necessary. Therefore, if you're going to attach a connector to other shapes, the position and dimensions you specify when adding the connector are irrelevant.

AddConnector Method Example

This example adds two rectangles to `myDocument` and connects them with a curved connector. Note that when you attach the connector to the rectangles, the size and position of the connector are automatically adjusted; therefore, the position and dimensions you specify when adding the callout are irrelevant (dimensions must be nonzero).

```
Set myDocument = ActivePresentation.Slides(1)
Set s = myDocument.Shapes
Set firstRect = s.AddShape(msoShapeRectangle, 100, 50, 200, 100)
Set secondRect = s.AddShape(msoShapeRectangle, 300, 300, 200, 100)
With s.AddConnector(msoConnectorCurve, 0, 0, 100, 100).ConnectorFormat
    .BeginConnect ConnectedShape:=firstRect, ConnectionSite:=1
    .EndConnect ConnectedShape:=secondRect, ConnectionSite:=1
    .Parent.RerouteConnections
End With
```

AddCurve Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddCurveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddCurveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddCurveA"}

Creates a Bézier curve. Returns a **Shape** object that represents the new curve.

Syntax

expression.**AddCurve(*SafeArrayOfPoints*)**

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.

AddCurve Method Example

The following example adds a two-segment Bézier curve to `myDocument`.

```
Dim pts(1 To 7, 1 To 2) As Single
pts(1, 1) = 0
pts(1, 2) = 0
pts(2, 1) = 72
pts(2, 2) = 72
pts(3, 1) = 100
pts(3, 2) = 40
pts(4, 1) = 20
pts(4, 2) = 50
pts(5, 1) = 90
pts(5, 2) = 120
pts(6, 1) = 60
pts(6, 2) = 30
pts(7, 1) = 150
pts(7, 2) = 90
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddCurve pts
```

AddLabel Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddLabelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddLabelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddLabelA"}

Creates a label. Returns a **Shape** object that represents the new label.

Syntax

expression.**AddLabel(*Orientation, Left, Top, Width, Height*)**

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

Orientation Required **Long**. The text orientation within the label. Can be either of the following **MsoTextOrientation** constants: **msoTextOrientationHorizontal** or **msoTextOrientationVerticalFarEast**. (Don't use any **MsoTextOrientation** constants other than these in U.S. English versions of PowerPoint.)

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the label relative to the upper-left corner of the document.

Width, ***Height*** Required **Single**. The width and height of the label, in points.

AddLabel Method Example

This example adds a vertical label that contains the text "Test Label" to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddLabel(msoTextOrientationVerticalFarEast, 100, 100, 60,
150) _
    .TextFrame.TextRange.Text = "Test Label"
```

AddLine Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddLineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddLineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddLineA"}

Creates a line. Returns a **Shape** object that represents the new line.

Syntax

expression.**AddLine(*BeginX, Beginy, EndX, EndY*)**

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

BeginX, ***BeginY*** Required **Single**. The position (in points) of the line's starting point relative to the upper-left corner of the document.

EndX, ***EndY*** Required **Single**. The position (in points) of the line's end point relative to the upper-left corner of the document.

AddLine Method Example

This example adds a blue dashed line to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddLine(10, 10, 250, 250).Line
    .DashStyle = msoLineDashDotDot
    .ForeColor.RGB = RGB(50, 0, 128)
End With
```

AddOLEObject Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddOLEObjectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddOLEObjectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddOLEObjectA"}

Creates an OLE object. Returns a **Shape** object that represents the new OLE object.

Syntax

expression.**AddOLEObject(*Left, Top, Width, Height, ClassName, FileName, DisplayAsIcon, IconFileName, IconIndex, IconLabel, Link*)**

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

Left, ***Top*** Optional **Single**. The position (in points) of the upper-left corner of the new object relative to the upper-left corner of the slide. The default value is 0 (zero).

Width, ***Height*** Optional **Single**. The initial dimensions of the OLE object, in points.

ClassName Optional **String**. The OLE long class name or the ProgID for the object that's to be created. You must specify either the ***ClassName*** or ***FileName*** argument for the object, but not both. For more information about programmatic identifiers, see OLE Programmatic Identifiers.

FileName Optional **String**. The file from which the object is to be created. If the path isn't specified, the current working folder is used. You must specify either the ***ClassName*** or ***FileName*** argument for the object, but not both.

DisplayAsIcon Optional **Long**. **True** to display the OLE object as an icon. The default value is **False**.

IconFileName Optional **String**. The file that contains the icon to be displayed.

IconIndex Optional **Long**. The index 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 (accessed from the **Insert 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 **String**. A label (caption) to be displayed beneath the icon.

Link Optional **Long**. **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 ***ClassName***, this argument must be **False**. The default value is **False**.

AddOLEObject Method Example

This example adds a linked Word document to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddOLEObject Left:=100, Top:=100, Width:=200,
Height:=300, _
    FileName:="c:\my documents\testing.doc", link:=True
```

This example adds a new Microsoft Excel worksheet to `myDocument`. The worksheet will be displayed as an icon.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddOLEObject Left:=100, Top:=100, Width:=200,
Height:=300, _
    ClassName:="Excel.Sheet.8", DisplayAsIcon:=True
```

This example adds a command button to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddOLEObject Left:=100, Top:=100, Width:=150, Height:=50,
_
    ClassName:="Forms.CommandButton.1"
```

AddPicture Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPictureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPictureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPictureA"}

Creates a picture from an existing file. Returns a **Shape** object that represents the new picture.

Syntax

expression.**AddPicture(*FileName, LinkToFile, SaveWithDocument, Left, Top, Width, Height*)**

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

FileName Required **String**. The file from which the OLE object is to be created.

LinkToFile Required **Long**. **True** to link the picture to the file from which it was created. **False** to make the picture an independent copy of the file.

SaveWithDocument Required **Long**. **True** to save the linked picture with the document into which it's inserted. **False** to storeonly the link information in the document. This argument must be **True** if ***LinkToFile*** is **False**.

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the picture relative to the upper-left corner of the document.

Width, ***Height*** Required **Single**. The width and height of the picture, in points.

AddPicture Method Example

This example adds a picture created from the file Music.bmp to `myDocument`. The inserted picture is linked to the file from which it was created and is saved with `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddPicture "c:\microsoft office\clipart\music.bmp", _
    True, True, 100, 100, 70, 70
```

AddPolyline Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPolylineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPolylineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPolylineA"}

Creates an open polyline or a closed polygon drawing. Returns a **Shape** object that represents the new polyline or polygon.

Syntax

expression.**AddPolyline(*SafeArrayOfPoints*)**

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

SafeArrayOfPoints Required **Variant**. An array of coordinate pairs that specifies the polyline drawing's vertices.

Remarks

To form a closed polygon, assign the same coordinates to the first and last vertices in the polyline drawing.

AddPolyline Method Example

This example adds a triangle to `myDocument`. Because the first and last points have the same coordinates, the polygon is closed and filled. The color of the triangle's interior will be the same as the default shape's fill color.

```
Dim triArray(1 To 4, 1 To 2) As Single
triArray(1, 1) = 25
triArray(1, 2) = 100
triArray(2, 1) = 100
triArray(2, 2) = 150
triArray(3, 1) = 150
triArray(3, 2) = 50
triArray(4, 1) = 25     ' Last point has same coordinates as first
triArray(4, 2) = 100
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddPolyline triArray
```

AddShape Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddShapeA"}

Creates an AutoShape. Returns a **Shape** object that represents the new AutoShape.

Syntax

expression.**AddShape(*Type, Left, Top, Width, Height*)**

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

Type Required **Long**. Specifies the type of AutoShape to create. Can be any one of the **MsoAutoShapeType** constants.

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the AutoShape's bounding box relative to the upper-left corner of the document.

Width, ***Height*** Required **Single**. The width and height of the AutoShape's bounding box, in points.

Remarks

To change the type of an AutoShape that you've added, set the **AutoShapeType** property.

AddShape Method Example

This example adds a rectangle to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddShape msoShapeRectangle, 50, 50, 100, 200
```

AddTextbox Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTextboxC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTextboxX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTextboxA"}

Creates a text box. Returns a **Shape** object that represents the new text box.

Syntax

expression.**AddTextbox(*Orientation, Left, Top, Width, Height*)**

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

Orientation Required **Long**. The text orientation within the label. Can be either of the following **MsoTextOrientation** constants: **msoTextOrientationHorizontal** or **msoTextOrientationVerticalFarEast**. (Don't use any **MsoTextOrientation** constant other than these in U.S. English versions of PowerPoint.)

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the text box relative to the upper-left corner of the document.

Width, ***Height*** Required **Single**. The width and height of the text box, in points.

AddTextbox Method Example

This example adds a text box that contains the text "Test Box" to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.AddTextbox(msoTextOrientationHorizontal, 100, 100, 200,
50) _
    .TextFrame.TextRange.Text = "Test Box"
```

AddTextEffect Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTextEffectA"}

Creates a WordArt object. Returns a **Shape** object that represents the new WordArt object.

Syntax

expression.**AddTextEffect(*PresetTextEffect, Text, FontName, FontSize, FontBold, FontItalic, Left, Top*)**

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

PresetTextEffect Required **Long**. The preset text effect. Can be one of the following **MsoPresetTextEffect** constants:

msoTextEffect1	**msoTextEffect23**
msoTextEffect10	**msoTextEffect24**
msoTextEffect11	**msoTextEffect25**
msoTextEffect12	**msoTextEffect26**
msoTextEffect13	**msoTextEffect27**
msoTextEffect14	**msoTextEffect28**
msoTextEffect15	**msoTextEffect29**
msoTextEffect16	**msoTextEffect3**
msoTextEffect17	**msoTextEffect30**
msoTextEffect18	**msoTextEffect4**
msoTextEffect19	**msoTextEffect5**
msoTextEffect2	**msoTextEffect6**
msoTextEffect20	**msoTextEffect7**
msoTextEffect21	**msoTextEffect8**
msoTextEffect22	**msoTextEffect9**

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 **Long**. **True** to set the font used in the WordArt to bold.

FontItalic Required **Long**. **True** to set the font used in the WordArt to italic.

Left, ***Top*** Required **Single**. The position (in points) of the upper-left corner of the WordArt's bounding box relative to the upper-left corner of the document.

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.

AddTextEffect Method Example

This example adds WordArt that contains the text "Test" to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
Set newWordArt =
myDocument.Shapes.AddTextEffect(PresetTextEffect:=msoTextEffect1,
Text:="Test", _
    FontName:="Arial Black", FontSize:=36, FontBold:=False,
FontItalic:=False, Left:=10, Top:=10)
```

AddTitle Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTitleA"}

Restores a previously deleted title placeholder to a slide. Returns a **Shape** object that represents the restored title.

Note This method will cause an error if you haven't previously deleted the title placeholder from the specified slide. Use the **HasTitle** property to determine whether the title placeholder has been deleted.

Syntax

expression.**AddTitle**

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

AddTitle Method Example

This example restores the title placeholder to slide one in the active presentation if this placeholder has been deleted. The text of the restored title is "Restored title."

```
With ActivePresentation.Slides(1)
    If .Layout <> ppLayoutBlank Then
        With .Shapes
            If Not .HasTitle Then
                .AddTitle.TextFrame.TextRange.Text = "Restored title"
            End If
        End With
    End If
End With
```

BuildFreeform Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthBuildFreeformC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthBuildFreeformX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthBuildFreeformA"}

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.

Syntax

expression.**BuildFreeform(*EditingType, X1, Y1*)**

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

EditingType Required **Long**. The editing property of the first node. Can be either of the following **MsoEditingType** constants: **msoEditingAuto** or **msoEditingCorner** (cannot be **msoEditingSmooth** or **msoEditingSymmetric**).

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.

BuildFreeform Method Example

This example adds a freeform with with four segments to `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

default shape

A virtual shape that contains default formatting properties for new shapes. There's one default shape for each presentation.

HasTitle Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro HasTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro HasTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro HasTitleA"}

True if the collection of objects on the specified slide contains a title placeholder. Read-only **Long**.

HasTitle Property Example

This example restores the title placeholder to slide one in the active presentation if this placeholder has been deleted. The text of the restored title is "Restored title."

```
With ActivePresentation.Slides(1)
    If .Layout <> ppLayoutBlank Then
        With .Shapes
            If Not .HasTitle Then
                .AddTitle.TextFrame.TextRange.Text = "Restored title"
            End If
        End With
    End If
End With
```

Placeholders Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPlaceholdersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPlaceholdersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPlaceholdersA"}

Returns a **Placeholders** collection that represents the collection of all the placeholders on a slide. Each placeholder in the collection can contain text, a chart, a table, an organizational chart, or another object. Read-only.

Placeholders Property Example

This example adds a slide to the active presentation and then adds text to both the title (which is the first placeholder on the slide) and the subtitle.

```
Set myDocument = ActivePresentation.Slides(1)
With ActivePresentation.Slides.Add(1, ppLayoutTitle).Shapes.Placeholders
    .Item(1).TextFrame.TextRange.Text = "This is the title text"
    .Item(2).TextFrame.TextRange.Text = "This is subtitle text"
End With
```

Range Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRangeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRangeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRangeA"}

Shapes object: Returns a **ShapeRange** object that represents a subset of the shapes in a **Shapes** collection.

Slides object: Returns a **SlideRange** object that represents a subset of the slides in a **Slides** collection.

Syntax

expression.**Range(*Index*)**

expression Required. An expression that returns either a **Shapes** or **Slides** object.

Index Optional **Variant**. The individual shapes or slides that are to be included in the range. Can be an integer that specifies the index number of the shape or slide, a string that specifies the name of the shape or slide, or an array that contains either integers or strings. If this argument is omitted, the **Range** method returns all the objects in the specified collection.

Remarks

Although you can use the **Range** method to return any number of shapes or slides, it's simpler to use the **Item** method if you only want to return a single member of the collection. For example, `Shapes(1)` is simpler than `Shapes.Range(1)`, and `Slides(2)` is simpler than `Slides.Range(2)`.

To specify an array of integers or strings for ***Index***, you can use the **Array** function. For example, the following instruction returns two shapes specified by name.

```
Set myRange = myDocument.Shapes.Range(Array("Oval 4", "Rectangle 5"))
```

Range Method Example

This example sets the fill pattern for shapes one and three on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned
msoPatternHorizontalBrick
```

This example sets the fill pattern for the shapes named "Oval 4" and "Rectangle 5" on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
Set myRange = myDocument.Shapes.Range(Array("Oval 4", "Rectangle 5"))
myRange.Fill.Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for all shapes on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Range.Fill.Patterned msoPatternHorizontalBrick
```

This example sets the fill pattern for shape one on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
Set myRange = myDocument.Shapes.Range(1)
myRange.Fill.Patterned msoPatternHorizontalBrick
```

This example creates an array that contains all the AutoShapes on `myDocument`, uses that array to define a shape range, and then distributes all the shapes in that range horizontally.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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
```

This example sets the title color for slides one and three in `myDocument`.

```
Set mySlides = ActivePresentation.Slides.Range(Array(1, 3))
mySlides .ColorScheme.Colors(ppTitle).RGB = RGB(0, 255, 0)
```

This example sets the title color for the slides named "Slide6" and "Slide8" in `myDocument`.

```
Set mySlides = ActivePresentation.Slides.Range(Array("Slide6", "Slide8"))
mySlides .ColorScheme.Colors(ppTitle).RGB = RGB(0, 255, 0)
```

This example sets the title color for all the slides in the active presentation.

```
Set mySlides = ActivePresentation.Slides.Range
mySlides.ColorScheme.Colors(ppTitle).RGB = RGB(255, 0, 0)
```

This example creates an array that contains all the title slides in the active presentation, uses that array to define a slide range, and then sets the title color for all slides in that range.

```
Dim MyTitleArray() As Long
Set pSlides = ActivePresentation.Slides
ReDim MyTitleArray(1 To pSlides.Count)
For Each pSlide In pSlides
    If pSlide.Layout = ppLayoutTitle Then
        nCounter = nCounter + 1
        MyTitleArray(nCounter) = pSlide.SlideIndex
    End If
Next pSlide
ReDim Preserve MyTitleArray(1 To nCounter)

Set rngTitleSlides = ActivePresentation.Slides.Range(MyTitleArray)
rngTitleSlides.ColorScheme.Colors(ppTitle).RGB = RGB(255, 123, 99)
```

SelectAll Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSelectAllC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSelectAllX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSelectAllA"}

Selects all the shapes in the specified **Shapes** collection.

Syntax

expression.**SelectAll**

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

SelectAll Method Example

This example selects all the shapes on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.SelectAll
```

Title Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTitleC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTitleX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTitleA"}

Returns a **Shape** object that represents the slide title. Read-only.

Remarks

You can also use the **Item** method of the **Shapes** or **Placeholders** collection to return the slide title.

Title Property Example

This example sets the title text on slide one in `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes.Title.TextFrame.TextRange.Text = "Welcome!"
```

EditingType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEditingTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEditingTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEditingTypeA"}

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. Can be one of the following **MsoEditingType** constants: **msoEditingAuto**, **msoEditingCorner**, **msoEditingSmooth**, or **msoEditingSymmetric**. If the node is a control point for a curved segment, this property returns the editing type of the adjacent vertex. Read-only **Long**.

Remarks

This property is read-only. Use the **SetEditingType** method to set the value of this property.

EditingType Property Example

This example changes all corner nodes to smooth nodes in shape three on `myDocument`. Shape three must be a freeform drawing.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Nodes
    For n = 1 to .Count
        If .Item(n).EditingType = msoEditingCorner Then
            .SetEditingType n, msoEditingSmooth
        End If
    Next
End With
```

Points Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro PointsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro PointsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro PointsA"}

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 value of this property.

coordinate pair

A pair of values representing the x- and y-coordinates of a point that are stored in a one-based two-dimensional array that can contain coordinates for many points.

The following example creates a one-based array that contains the coordinate pairs 50, 150 and 100, 200. Note that when you dimension the array, the size of the first dimension is the number of points you want to include, and the size of the second dimension is two (the first position within the dimension is for the x-coordinate, the second is for the y-coordinate).

```
Dim cpArray(1 To 2, 1 To 2)
cpArray(1, 1) = 50' The x-coordinate of the first point
cpArray(1, 2) = 150  ' The y-coordinate of the first point
cpArray(2, 1) = 100  ' The x-coordinate of the second point
cpArray(2, 2) = 200  ' The y-coordinate of the second point
```

Points Property 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 = ActivePresentation.Slides(1)
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
```

SegmentType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproSegmentTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproSegmentTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproSegmentTypeA"}

Returns a value that indicates whether the segment assocated with the specified node is straight or curved. Can be either of the following **MsoSegmentType** constants: **msoSegmentCurve** or **msoSegmentLine**. If the specified node is a control point for a curved segment, this property returns **msoSegmentCurve**. Read-only **Long**.

Remarks

This property is read-only. Use the **SetSegmentType** method to set the value of this property.

SegmentType 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 = ActivePresentation.Slides(1)
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
```

Insert Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertA"}

Inserts a new segment after the specified node of the freeform.

Syntax

expression.**Insert(*Index, SegmentType, EditingType, X1, Y1, X2, Y2, X3, Y3*)**

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

Index Required **Long**. The node that the new node is to be inserted after.

SegmentType Required **Long**. The type of segment to be added. Can be either of the following **MsoSegmentType** constants: **msoSegmentCurve** or **msoSegmentLine**.

EditingType Required **Long**. The editing property of the vertex. Can be either of the following **MsoEditingType** constants: **msoEditingAuto** or **msoEditingCorner** (cannot be **msoEditingSmooth** or **msoEditingSymmetric**).

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.

Insert Method Example

This example adds a smooth node with a curved segment after node four in shape three on `myDocument`. Shape three must be a freeform drawing with at least four nodes.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Nodes
    .Insert 4, msoSegmentCurve, msoEditingSmooth, 210, 100
End With
```

SetEditingType Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetEditingTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetEditingTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetEditingTypeA"}

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.

Syntax

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 **Long**. The editing property of the vertex. Can be one of the following **MsoEditingType** constants: **msoEditingAuto**, **msoEditingCorner**, **msoEditingSmooth**, or **msoEditingSymmetric**.

SetEditingType Method Example

This example changes all corner nodes to smooth nodes in shape three on `myDocument`. Shape three must be a freeform drawing.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Nodes
    For n = 1 to .Count
        If .Item(n).EditingType = msoEditingCorner Then
            .SetEditingType n, msoEditingSmooth
        End If
    Next
End With
```

SetPosition Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetPositionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetPositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetPositionA"}

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.

Syntax

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.

SetPosition Method 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(3).Nodes
    pointsArray = .Item(2).Points
    currXvalue = pointsArray(0, 0)
    currYvalue = pointsArray(0, 1)
    .SetPosition 2, currXvalue + 200, currYvalue + 300
End With
```

SetSegmentType Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetSegmentTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetSegmentTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetSegmentTypeA"}

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.

Syntax

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 **Long**. Specifies if the segment is straight or curved. Can be either of the following **MsoSegmentType** constants: **msoSegmentCurve** or **msoSegmentLine**.

SetSegmentType Method Example

This example changes all straight segments to curved segments in shape three on `myDocument`. Shape three must be a freeform drawing.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

Background Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BackgroundA"}

Master, **Slide**, and **SlideRange** objects: Returns a **ShapeRange** object that represents the slide background. Read-only.

Remarks

If you want to use the **Background** property to set the background for an individual slide without changing the slide master, the **FollowMasterBackground** property for that slide must be set to **False**.

Background Property Example

This example sets the background of the slide master in the active presentation to a preset shade.

```
ActivePresentation.SlideMaster.Background.Fill.PresetGradient _
    msoGradientHorizontal, 1, msoGradientLateSunset
```

This example sets the background of slide one in the active presentation to a preset shade.

```
With ActivePresentation.Slides(1)
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

FollowMasterBackground Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFollowMasterBackgroundC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFollowMasterBackgroundX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFollowMasterBackgroundA"}

True if the specified slide or range of slides follows the slide master background. **False** if the specified slide or range of slides has a custom background. Read/write **Long**.

Remarks

When you create a new slide, the default value for this property is **True**. If you copy a slide from another presentation, it retains the setting it had in the original presentation. That is, if the slide followed the slide master background in the original presentation, it will automatically follow the slide master background in the new presentation; or, if the slide had a custom background, it will retain that custom background.

Note that the look of the slide's background is determined by the color scheme and background objects as well as by the background itself. If setting the **FollowMasterBackground** property alone doesn't give you the results you want, try setting the **ColorScheme** and **DisplayMasterShapes** properties as well.

FollowMasterBackground Property Example

This example copies slide one from presentation two, pastes the slide at the end of presentation one, and matches the slide's background, color scheme, and background objects to the rest of presentation one.

```
Presentations(2).Slides(1).Copy
With Presentations(1).Slides.Paste
    .FollowMasterBackground = True
    .ColorScheme = Presentations(1).SlideMaster.ColorScheme
    .DisplayMasterShapes = True
End With
```

DisplayMasterShapes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDisplayMasterShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDisplayMasterShapesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDisplayMasterShapesA"}

True if the specified slide or range of slides displays the background objects on the slide master. These background objects can include text, drawings, OLE objects, and clip art you add to the slide master. Headers and footers aren't included. Read/write **Long**.

Remarks

When you create a new slide, the default value for this property is **True**. If you copy a slide from another presentation, it retains the setting it had in the original presentation. That is, if the slide omitted slide master background objects in the original presentation, it will omit them in the new presentation as well.

Note that the look of the slide's background is determined by the color scheme and background as well as by the background objects. If setting the **DisplayMasterShapes** property alone doesn't give you the results you want, try setting the **FollowMasterBackground** and **ColorScheme** properties as well.

DisplayMasterShapes Property Example

This example copies slide one from presentation two, pastes it at the end of presentation one, and matches the slide's background, color scheme, and background objects to the rest of presentation one.

```
Presentations(2).Slides(1).Copy
With Presentations(1).Slides.Paste
    .FollowMasterBackground = True
    .ColorScheme = Presentations(1).SlideMaster.ColorScheme
    .DisplayMasterShapes = True
End With
```

HeadersFooters Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHeadersFootersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHeadersFootersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHeadersFootersA"}

Returns a **HeadersFooters** collection that represents the header, footer, date and time, and slide number associated with the slide, slide master, or range of slides. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

HeadersFooters Property Example

This example sets the footer text and the date and time format for the notes master in the active presentation and sets the date and time to be updated automatically, .

```
With ActivePresentation.NotesMaster.HeadersFooters
    .Footer.Text = "Regional Sales"
    With .DateAndTime
        .UseFormat = True
        .Format = ppDateTimeHmmss
    End With
End With
```

Hyperlinks Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHyperlinksC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHyperlinksX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHyperlinksA"}

Returns a **Hyperlinks** collection that represents all the hyperlinks on the specified slide. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Hyperlinks Property Example

This example allows the user to update an outdated internet address for all hyperlinks in the the active presentation.

```
oldAddr = InputBox("Old internet address")
newAddr = InputBox("New internet address")
For Each s In ActivePresentation.Slides
    For Each h In s.Hyperlinks
        If LCase(h.Address) = oldAddr Then h.Address = newAddr
    Next
Next
```

Layout Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLayoutC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLayoutX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLayoutA"}

Returns or sets the slide layout. Read/write **Long**.

Can be one of the following **PpSlideLayout** constants.

ppLayoutBlank
ppLayoutChart
ppLayoutChartAndText
ppLayoutClipartAndText
ppLayoutClipArtAndVerticalText
ppLayoutFourObjects
ppLayoutLargeObject
ppLayoutMediaClipAndText
ppLayoutMixed
ppLayoutObject
ppLayoutObjectAndText
ppLayoutObjectOverText
ppLayoutOrgchart
ppLayoutTable
ppLayoutText
ppLayoutTextAndChart
ppLayoutTextAndClipart
ppLayoutTextAndMediaClip
ppLayoutTextAndObject
ppLayoutTextAndTwoObjects
ppLayoutTextOverObject
ppLayoutTitle
ppLayoutTitleOnly
ppLayoutTwoColumnText
ppLayoutTwoObjectsAndText
ppLayoutTwoObjectsOverText
ppLayoutVerticalText
ppLayoutVerticalTitleAndText
ppLayoutVerticalTitleAndTextOverChart

Layout Property Example

This example changes the layout of slide one in the active presentation to include a title and subtitle if it initially has only a title.

```
With ActivePresentation.Slides(1)
    If .Layout = ppLayoutTitleOnly Then
        .Layout = ppLayoutTitle
    End If
End With
```

Master Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproMasterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproMasterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproMasterA"}

Returns a **Master** object that represents the slide master. Read-only.

Master Property Example

This example sets the background fill for the slide master for slide one in the active presentation.

```
ActivePresentation.Slides(1).Master.Background.Fill _
    .PresetGradient msoGradientDiagonalUp, 1, msoGradientDaybreak
```

NotesPage Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproNotesPageC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproNotesPageX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproNotesPageA"}

Returns a **SlideRange** object that represents the notes pages for the specified slide or range of slides. Read-only.

Note The following properties and methods will fail if applied to a **SlideRange** object that represents a notes page: **Copy** method, **Cut** method, **Delete** method, **Duplicate** method, **HeadersFooters** property, **Hyperlinks** property, **Layout** property, **PrintSteps** property, **SlideShowTransition** property.

Remarks

The **NotesPage** property returns the notes page for either a single slide or a range of slides and allows you to make changes only to those notes pages. If you want to make changes that affect all notes pages, use the **NotesMaster** property to return the **Slide** object that represents the notes master.

NotesPage Property Example

This example sets the background fill for the notes page for slide one in the active presentation.

```
With ActivePresentation.Slides(1).NotesPage
    .FollowMasterBackground = False
    .Background.Fill.PresetGradient msoGradientHorizontal, 1,
msoGradientLateSunset
End With
```

PrintSteps Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPrintStepsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPrintStepsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPrintStepsA"}

Returns the number of slides you'd need to print to simulate the builds on the specified slide, slide master, or range of slides. Read-only **Long**.

PrintSteps Property Example

This example sets a variable to the number of slides you'd need to print to simulate the builds on slide one in the active presentation and then displays the value of the variable.

```
steps1 = ActivePresentation.Slides(1).PrintSteps
MsgBox steps1
```

Select Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSelectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSelectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSelectA"}

Selects the specified object.

Syntax 1

expression.**Select**

expression Required. An expression that returns a **Slide**, **SlideRange**, or **TextRange** object.

Syntax 2

expression.**Select(*Replace*)**

expression Required. An expression that returns a **Shape** or **ShapeRange** object.

Replace Optional **Long**. **True** if the selection replaces any previous selection. **False** if the selection is added to the previous selection. **True** by default.

Remarks

If you try to make a selection that isn't appropriate for the view, your code will fail. For example, you can select a slide in slide sorter view but not in slide view.

Select Method Example

This example selects shapes one and three on slide one in the active presentation.

```
ActivePresentation.Slides(1).Shapes.Range(Array(1, 3)).Select
```

This example adds shapes two and and four on slide one in the the active presentation to the previous selection.

```
ActivePresentation.Slides(1).Shapes.Range(Array(2, 4)).Select False
```

This example selects the first five characters in the title of slide one in the the active presentation.

```
ActivePresentation.Slides(1).Shapes.Title.TextFrame.TextRange.Characters(1,
5).Select
```

This example selects slide one in the active presentation.

```
ActivePresentation.Slides(1).Select
```

Shapes Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShapesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShapesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShapesA"}

Returns a **Shapes** collection that represents all the elements that have been placed or inserted on the specified slide, slide master, or range of slides. This collection can contain the drawings, shapes, OLE objects, pictures, text objects, titles, headers, footers, slide numbers, and date and time objects on a slide, or on the slide image on a notes page. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

Shapes Property Example

This example adds a rectangle that's 100 points wide and 50 points high, and whose upper-left corner is 5 points from the left edge of slide one in the active presentation and 25 points from the top of the slide.

```
Set firstSlide = ActivePresentation.Slides(1)
firstSlide.Shapes.AddShape msoShapeRectangle, 5, 25, 100, 50
```

This example sets the fill texture for shape three on slide one in the active presentation.

```
Set newRect = ActivePresentation.Slides(1).Shapes(3)
newRect.Fill.PresetTextured msoTextureOak
```

Assuming that slide one in the active presentation contains a title, both the second and third lines of code in the following example set the title text on slide one in the presentation.

```
Set firstSl = ActivePresentation.Slides(1)
firstSl.Shapes.Title.TextFrame.TextRange.Text = "Some title text"
firstSl.Shapes(1).TextFrame.TextRange.Text = "Other title text"
```

Assuming that shape two on slide two in the active presentation contains a text frame, the following example adds a series of paragraphs to the slide. Note that `Chr(13)` is used to insert paragraph marks within the text.

```
Set tShape = ActivePresentation.Slides(2).Shapes(2)
tShape.TextFrame.TextRange.Text = "First Item" & Chr(13) & _
    "Second Item" & Chr(13) & "Third Item"
```

For most slide layouts, the first shapes on the slide are text placeholders, and the following example accomplishes the same task as the preceding example.

```
Set testShape = ActivePresentation.Slides(2).Shapes.Placeholders(2)
testShape.TextFrame.TextRange.Text = "First Item" & _
    Chr(13) & "Second Item" & Chr(13) & "Third Item"
```

SlideID Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideIDC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideIDX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideIDA"}

Returns a unique ID number for the specified slide. Read-only **Long**.

Remarks

Unlike the **SlideIndex** property, the **SlideID** property of a **Slide** object won't change when you add slides to the presentation or rearrange the slides in the presentation. Therefore, using the **FindBySlideID** method with the slide's ID number can be a more reliable way to return a specific **Slide** object from a **Slides** collection than using the **Item** method with the slide's index number.

SlideID Property Example

This example demonstrates how to retrieve the unique ID number for a **Slide** object and then use this number to return that **Slide** object from the **Slides** collection.

```
Set gslides = ActivePresentation.Slides
graphSlideID = gslides.Add(2, ppLayoutChart).SlideID    'Get slide ID
gslides.FindBySlideID(graphSlideID).SlideShowTransition.EntryEffect = _
    ppEffectCoverLeft      'Use ID to return specific slide
```

SlideIndex Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideIndexC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideIndexX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideIndexA"}

Returns the index number of the specified slide within the **Slides** collection. Read-only **Long**.

Remarks

Unlike the **SlideID** property, the **SlideIndex** property of a **Slide** object can change when you add slides to the presentation or rearrange the slides in the presentation. Therefore, using the **FindBySlideID** method with the slide's ID number can be a more reliable way to return a specific **Slide** object from a **Slides** collection than using the **Item** method with the slide's index number.

SlideIndex Property Example

This example displays the index number of the currently displayed slide in slide show window one.

```
MsgBox SlideShowWindows(1).View.Slide.SlideIndex
```

SlideShowTransition Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideShowTransitionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideShowTransitionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideShowTransitionA"}

Returns a **SlideShowTransition** object that represents the special effects for the specified slide transition. Read-only.

SlideShowTransition Property Example

This example sets slide two in the active presentation to advance automatically after 5 seconds during a slide show and to play a dog bark sound at the slide transition.

```
With ActivePresentation.Slides(2).SlideShowTransition
    .AdvanceOnTime = True
    .AdvanceTime = 5
    .SoundEffect.ImportFromFile "c:\windows\media\dogbark.wav"
End With
ActivePresentation.SlideShowSettings.AdvanceMode = _
    ppSlideShowUseSlideTimings
```

Add Method (Slides Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddSlidesObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddSlidesObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddSlidesObjA"}

Creates a new slide and adds it to the collection of slides in the specified presentation. Returns a **Slide** object that represents the new slide.

Syntax

expression.**Add(*Index, Layout*)**

expression Required. An expression that returns a **Slides** collection.

Index Required **Long**. Specifies the index number the new slide will have within the **Slides** collection. This value cannot exceed the number of existing slides + 1.

Layout Required **Long**. Can be one of the following **PpSlideLayout** constants:

ppLayoutBlank	**ppLayoutTextAndChart**
ppLayoutChart	**ppLayoutTextAndClipart**
ppLayoutChartAndText	**ppLayoutTextAndMediaClip**
ppLayoutClipartAndText	**ppLayoutTextAndObject**
ppLayoutClipArtAndVerticalText	**ppLayoutTextAndTwoObjects**
ppLayoutFourObjects	**ppLayoutTextOverObject**
ppLayoutLargeObject	**ppLayoutTitle**
ppLayoutMediaClipAndText	**ppLayoutTitleOnly**
ppLayoutObject	**ppLayoutTwoColumnText**
ppLayoutObjectAndText	**ppLayoutTwoObjectsAndText**
ppLayoutObjectOverText	**ppLayoutTwoObjectsOverText**
ppLayoutOrgchart	**ppLayoutVerticalText**
ppLayoutTable	**ppLayoutVerticalTitleAndText**
ppLayoutText	**ppLayoutVerticalTitleAndTextOverChart**

Remarks

To alter the layout of an existing slide, use the **Layout** property.

Add Method (Slides Collection Object) Example

This example adds a slide that contains a title placeholder at the beginning of the active presentation.

```
ActivePresentation.Slides.Add 1, ppLayoutTitleOnly
```

This example adds a blank slide at the end of the active presentation.

```
With ActivePresentation.Slides
    .Add .Count + 1, ppLayoutBlank
End With
```

FindBySlideID Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFindBySlideIDC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFindBySlideIDX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFindBySlideIDA"}

Returns a **Slide** object that represents the slide with the specified slide ID number. Each slide is automatically assigned a unique slide ID number when it's created. Use the **SlideID** property to return a slide's ID number.

Syntax

expression.**FindBySlideID(*SlideID*)**

expression Required. An expression that returns a **Slides** collection.

SlideID Required **Long**. Specifies the ID number of the slide you want to return. PowerPoint assigns this number when the slide is created.

Remarks

Unlike the **SlideIndex** property, the **SlideID** property of a **Slide** object won't change when you add slides to the presentation or rearrange the slides in the presentation. Therefore, using the **FindBySlideID** method with the slide ID number can be a more reliable way to return a specific **Slide** object from a **Slides** collection than using the **Item** method with the slide's index number.

FindBySlideID Method Example

This example demonstrates how to retrieve the unique ID number for a **Slide** object and then use this number to return that **Slide** object from the **Slides** collection.

```
Set gslides = ActivePresentation.Slides
graphSlideID = gslides.Add(2, ppLayoutChart).SlideID    'Get slide ID
gslides.FindBySlideID(graphSlideID).SlideShowTransition.EntryEffect = _
    ppEffectCoverLeft      'Use ID to return specific slide
```

InsertFromFile Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertFromFileC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertFromFileX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertFromFileA)"}

Inserts slides from a file into a presentation, at the specified location. Returns an integer that represents the number of slides inserted.

Syntax

expression.**InsertFromFile(*FileName, Index, SlideStart, SlideEnd*)**

expression Required. An expression that returns a **Slides** collection.

FileName Required **String**. The name of the file that contains the slides you want to insert.

Index Required **Long**. The index number of the **Slide** object in the specified **Slides** collection you want to insert the new slides after.

SlideStart Optional **Long**. The index number of the first **Slide** object in the **Slides** collection in the file denoted by ***FileName***.

SlideEnd Optional **Long**. The index number of the last **Slide** object in the **Slides** collection in the file denoted by ***FileName***.

InsertFromFile Method Example

This example inserts slides three through six from C:\Ppt\Sales.ppt after slide two in the active presentation.

```
ActivePresentation.Slides.InsertFromFile "c:\ppt\sales.ppt", 2, 3, 6
```

TextStyles Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTextStylesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTextStylesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTextStylesA"}

Returns a **TextStyles** collection that represents three text styles – title text, body text, and default text – for the specified slide master. Read-only.

For information about returning a single member of a collection, see Returning an Object from a Collection.

TextStyles Property Example

This example sets the font name and font size for level-one body text on slides in the active presentation.

```
With ActivePresentation.SlideMaster.TextStyles(ppBodyStyle).Levels(1)
    With .Font
        .Name = "arial"
        .Size = 36
    End With
End With
```

EndingSlide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproEndingSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproEndingSlideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproEndingSlideA"}

Returns or sets the last slide to be displayed in the specified slide show. Read/write **Long**.

EndingSlide Property Example

This example runs a slide show of the active presentation, starting with slide two and ending with slide four.

```
With ActivePresentation.SlideShowSettings
    .StartingSlide = 2
    .EndingSlide = 4
    .Run
End With
```

NamedSlideShows Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproNamedSlideShowsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproNamedSlideShowsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproNamedSlideShowsA"}

Returns a **NamedSlideShows** collection that represents all the named slide shows (custom slide shows) in the specified presentation. Each named slide show, or custom slide show, is a user-selected subset of the specified presentation. Read-only.

Remarks

Use the **Add** method of the **NamedSlideShows** object to create a named slide show.

NamedSlideShows Property Example

This example adds to the active presentation a named slide show "Quick Show" that contains slides 2, 7, and 9. The example then runs this slide show.

```
Dim qSlides(1 To 3) As Long
With ActivePresentation
    With .Slides
        qSlides(1) = .Item(2).SlideID
        qSlides(2) = .Item(7).SlideID
        qSlides(3) = .Item(9).SlideID
    End With
    With .SlideShowSettings
        .RangeType = ppShowNamedSlideShow
        .NamedSlideShows.Add "Quick Show", qSlides
        .SlideShowName = "Quick Show"
        .Run
    End With
End With
```

PointerColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPointerColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPointerColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPointerColorA"}

SlideShowSettings object: Returns the pointer color for the specified presentation as a **ColorFormat** object. This color is saved with the presentation and is the default pen color each time you show the presentation. Read-only.

SlideShowView object: Returns a **ColorFormat** object that represents the pointer color for the specified presentation during one slide show. As soon as the slide show is finished, the color reverts to the default color for the presentation. Read-only.

Remarks

To change the pointer to a pen, set the **PointerType** property to **ppSlideShowPointerPen**.

PointerColor Property Example

This example sets the default pen color for the active presentation to blue, starts a slide show, changes the pointer to a pen, and then sets the pen color to red for this slide show only.

```
With ActivePresentation.SlideShowSettings
    .PointerColor.RGB = RGB(0, 0, 255)          'blue
    With .Run.View
        .PointerColor.RGB = RGB(255, 0, 0)      'red
        .PointerType = ppSlideShowPointerPen
    End With
End With
```

Run Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRunC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRunX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRunA"}

Application object: Runs a Visual Basic procedure.

SlideShowSettings object: Runs a slide show of the specified presentation. Returns a **SlideShowWindow** object.

Syntax 1

expression.**Run(*MacroName, safeArrayOfParams*)**

Syntax 2

expression.**Run**

expression Required. An expression that returns an **Application** object (Syntax 1) or a **SlideShowSettings** object (Syntax 2).

MacroName Required **String**. The name of the procedure to be run. The string can contain the following: a loaded presentation or add-in file name followed by an exclamation point (!), a valid module name followed by a period (.), and the procedure name. For example, the following is a valid ***MacroName*** value: "MyPres.ppt!Module1.Test."

safeArrayOfParams Required **Variant**. The argument to be passed to the procedure. You cannot specify an object for this argument, and you cannot use named arguments with this method. Arguments must be passed by position.

Remarks

To run a custom slide show, set the **RangeType** property to **ppShowNamedSlideShow**, and set the **SlideShowName** property to the name of the custom show you want to run.

Run Method Example

This example starts a full-screen slide show of the active presentation, with shortcut keys disabled.

```
With ActivePresentation.SlideShowSettings
    .ShowType = ppShowSpeaker
    .Run.View.AcceleratorsEnabled = False
End With
```

This example runs the named slide show "Quick Show."

```
With ActivePresentation.SlideShowSettings
    .RangeType = ppShowNamedSlideShow
    .SlideShowName = "Quick Show"
    .Run
End With
```

In this example, the Main procedure defines an array and then runs the macro TestPass, passing the array as an argument.

```
Sub Main()
    Dim x(1 To 2)
    x(1) = "hi"
    x(2) = 7
    Application.Run "TestPass", x
End Sub

Sub TestPass(x)
    MsgBox x(1)
    MsgBox x(2)
End Sub
```

ShowType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShowTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShowTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShowTypeA"}

Returns or sets the show type for the specified slide show. Can be one of the following **PpSlideShowType** constants: **ppShowTypeKiosk**, **ppShowTypeSpeaker**, or **ppShowTypeWindow**. Read/write **Long**.

ShowType Property Example

This example runs a slide show of the active presentation in a window, starting with slide two and ending with slide four. The new slide show window is placed in the upper-left corner of the screen, and its width and height are both 300 points.

```
With ActivePresentation.SlideShowSettings
    .StartingSlide = 2
    .EndingSlide = 4
    .ShowType = ppShowTypeWindow
    With .Run
        .Left = 0
        .Top = 0
        .Width = 300
        .Height = 300
    End With
End With
```

ShowWithAnimation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShowWithAnimationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShowWithAnimationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShowWithAnimationA"}

True if the specified slide show displays shapes with assigned animation settings. Read/write **Long**.

ShowWithAnimation Property Example

This example runs a slide show of the active presentation with animation and narration turned off.

```
With ActivePresentation.SlideShowSettings
    .ShowWithAnimation = False
    .ShowWithNarration = False
    .Run
End With
```

ShowWithNarration Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproShowWithNarrationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproShowWithNarrationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproShowWithNarrationA"}

True if the specified slide show is shown with narration. Read/write **Long**.

ShowWithNarration Property Example

This example runs a slide show of the active presentation with animation and narration turned off.

```
With ActivePresentation.SlideShowSettings
    .ShowWithAnimation = False
    .ShowWithNarration = False
    .Run
End With
```

StartingSlide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproStartingSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproStartingSlideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproStartingSlideA"}

Returns or sets the first slide to be displayed in the specified slide show. Read/write **Long**.

StartingSlide Property Example

This example runs a slide show of the active presentation, starting with slide two and ending with slide four.

```
With ActivePresentation.SlideShowSettings
    .StartingSlide = 2
    .EndingSlide = 4
    .Run
End With
```

AdvanceOnClick Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAdvanceOnClickC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAdvanceOnClickX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAdvanceOnClickA"}

True if the specified slide advances when it's clicked during a slide show. Read/write **Long**.

Remarks

To set the slide to advance automatically after a certain amount of time elapses, set the **AdvanceOnTime** property to **True** and set the **AdvanceTime** property to the amount of time you want the slide to be shown. If you set both the **AdvanceOnClick** and the **AdvanceOnTime** properties to **True**, the slide will advance either when it's clicked or when the specified amount of time has elapsed – whichever comes first.

AdvanceOnClick Property Example

This example sets slide one in the active presentation to advance after five seconds have passed or when the mouse is clicked – whichever occurs first.

```
With ActivePresentation.Slides(1).SlideShowTransition
    .AdvanceOnClick = True
    .AdvanceOnTime = True
    .AdvanceTime = 5
End With
```

AdvanceOnTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAdvanceOnTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAdvanceOnTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAdvanceOnTimeA"}

True if the specified slide advances automatically after a specified amount of time has elapsed. Use the **AdvanceTime** property to specify the number of seconds after which the slide will automatically advance. Read/write **Long**.

Remarks

Set the **AdvanceMode** property of the **SlideShowSettings** object to **ppSlideShowUseSlideTimings** to put the slide interval settings into effect for the entire slide show.

AdvanceOnTime Property Example

This example sets slide one in the active presentation to advance after five seconds have passed or when the mouse is clicked – whichever occurs first.

```
With ActivePresentation.Slides(1).SlideShowTransition
    .AdvanceOnClick = True
    .AdvanceOnTime = True
    .AdvanceTime = 5
End With
```

Hidden Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHiddenC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHiddenX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHiddenA"}

True if the specified slide is hidden during a slide show. Read/write **Long**.

Hidden Property Example

This example makes slide two in the active presentation a hidden slide.

```
ActivePresentation.Slides(2).SlideShowTransition.Hidden = True
```

LoopSoundUntilNext Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproLoopSoundUntilNextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproLoopSoundUntilNextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproLoopSoundUntilNextA"}

True if the sound that's been set for the specified slide transition loops until the next sound starts. Read/write **Long**.

LoopSoundUntilNext Property Example

This example specifies that the file Dudududu.wav will start to play at the transition to slide two in the active presentation and will continue to play until the next sound starts.

```
With ActivePresentation.Slides(2).SlideShowTransition
    .SoundEffect.ImportFromFile "c:\sndsys\dudududu.wav"
    .LoopSoundUntilNext = True
End With
```

Speed Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSpeedC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSpeedX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSpeedA"}

Returns or sets the speed of the transition to the specified slide. Can be one of the following **PpTransitionSpeed** constants: **ppTransitionSpeedFast**, **ppTransitionSpeedMedium**, **ppTransitionSpeedMixed**, or **ppTransitionSpeedSlow**. Read/write **Long**.

Speed Property Example

This example sets the special effect for the transition to slide one in the active presentation to Strips Down-Left and specifies that the transition be fast.

```
With ActivePresentation.Slides(1).SlideShowTransition
    .EntryEffect = ppEffectStripsDownLeft
    .Speed = ppTransitionSpeedFast
End With
```

ImportFromFile Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthImportFromFileC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthImportFromFileX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthImportFromFileA"}

Specifies the sound that will be played whenever the specified shape is clicked or animated or whenever the specified slide transition occurs.

Syntax

expression.**ImportFromFile(*FullName*)**

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

FullName Required **String**. The name of the specified sound file.

ImportFromFile Method Example

This example specifies that the file Dudududu.wav will start to play at the transition to slide two in the active presentation and will continue to play until the next sound starts.

```
With ActivePresentation.Slides(2).SlideShowTransition
    .SoundEffect.ImportFromFile "c:\sndsys\dudududu.wav"
    .LoopUntilNextSound = True
End With
```

Play Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthPlayC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthPlayX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthPlayA"}

Plays the specified sound effect.

Syntax

expression.**Play**

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

Play Method Example

This example plays the sound effect that's been set for the transition to slide two in the active presentation.

```
ActivePresentation.Slides(2).SlideShowTransition.SoundEffect.Play
```

Clear Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthClearC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthClearX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthClearA"}

Clears the specified tab stop from the ruler and deletes it from the **TabStops** collection.

Clear Method Example

This example clears all tab stops for the text in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame _
    .Ruler.TabStops
    For i = .Count To 1 Step -1
        .Item(i).Clear
    Next
End With
```

Position Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro PositionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro PositionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro PositionA"}

Returns or sets the position of the specified tab stop, in points. Read/write **Single**.

Position Property Example

This example deletes all tab stops greater than 1 inch (72 points) for the text in shape two on slide one in the acive presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame _
    .Ruler.TabStops
    For i = .Count To 1 Step -1
        With .Item(i)
            If .Position > 72 Then .Clear
        End With
    Next
End With
```

Add Method (TabStops Collection Object)

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddTabStopsObjC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddTabStopsObjX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddTabStopsObjA"}

Adds a tab stop to the ruler for the specified text. Returns a **TabStop** object that represents the new tab stop.

Syntax

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

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

Type Required **Long**. Specifies the way text will be aligned with the new tab stop. Can be one of the following **PpTabStopType** constants: **ppTabStopCenter**, **ppTabStopDecimal**, **ppTabStopLeft**, or **ppTabStopRight**.

Position Required **Single**. The position of the new tab stop, in points.

Add Method (TabStops Collection Object) Example

This example sets a left-aligned tab stop at 2 inches (144 points) for the text in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame _
    .Ruler.TabStops.Add ppTabStopLeft, 144
```

DefaultSpacing Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDefaultSpacingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDefaultSpacingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDefaultSpacingA"}

Returns or sets the default tab-stop spacing for the specified text, in points. Read/write **Single**.

DefaultSpacing Property Example

This example sets the default tab-stop spacing to 0.5 inch (36 points) for the text in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame _
    .Ruler.TabStops.DefaultSpacing = 36
```

Name Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthNameA"}

Returns the name of the specified tag as a **String**.

Syntax

expression.**Name(*Index*)**

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

Index Required **Long**. The tag number.

Name Method Example

This example displays the name and value for each tag associated with slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Tags
    For i = 1 To .Count
        MsgBox "Tag #" & i & ": Name = " & .Name(i)
        MsgBox "Tag #" & i & ": Value = " & .Value(i)
    Next
End With
```

This example searches through the tags for each slide in the active presentation. If there's a tag named "PRIORITY," a message box displays the tag value. If there isn't a tag named "PRIORITY," the example adds this tag with the value "Unknown."

```
For Each s In Application.ActivePresentation.Slides
    With s.Tags
        found = False
        For i = 1 To .Count
            If .Name(i) = "PRIORITY" Then
                found = True
                slNum = .Parent.SlideIndex
                MsgBox "Slide " & slNum & " priority: " & .Value(i)
            End If
        Next
        If Not found Then
            slNum = .Parent.SlideIndex
            .Add "Name", "New Figures"
            .Add "Priority", "Unknown"
            MsgBox "Slide " & slNum & " priority tag added: Unknown"
        End If
    End With
Next
```

Value Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthValueC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthValueX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthValueA"}

Returns the value of the specified tag as a **String**.

Syntax

expression.**Value(*Index*)**

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

Index Required **Long**. The tag number.

Value Method Example

This example displays the name and value for each tag associated with slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Tags
    For i = 1 To .Count
        MsgBox "Tag #" & i & ": Name = " & .Name(i)
        MsgBox "Tag #" & i & ": Value = " & .Value(i)
    Next
End With
```

This example searches through the tags for each slide in the active presentation. If there's a tag named "PRIORITY," a message box displays the tag value. If there isn't a tag named "PRIORITY," the example adds this tag with the value "Unknown."

```
For Each s In Application.ActivePresentation.Slides
    With s.Tags
        found = False
        For i = 1 To .Count
            If .Name(i) = "PRIORITY" Then
                found = True
                slNum = .Parent.SlideIndex
                MsgBox "Slide " & slNum & " priority: " & .Value(i)
            End If
        Next
        If Not found Then
            slNum = .Parent.SlideIndex
            .Add "Name", "New Figures"
            .Add "Priority", "Unknown"
            MsgBox "Slide " & slNum & " priority tag added: Unknown"
        End If
    End With
Next
```

FontBold Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontBoldC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontBoldX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontBoldA"}

True if the font in the specified WordArt is bold. Read/write **Long**.

FontBold Property Example

This example sets the font to bold for shape three on `myDocument` if the shape is WordArt.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontBold = True
    End If
End With
```

FontItalic Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontItalicC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontItalicX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontItalicA"}

True if the font in the specified WordArt is italic. Read/write **Long**.

FontItalic Property Example

This example sets the font to italic for the shape named "WordArt 4" in `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes("WordArt 4").TextEffect.FontItalic = True
```

FontName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontNameA"}

Returns or sets the name of the font in the specified WordArt. Read/write **String**.

FontName Property Example

This example sets the font name to "Courier New" for shape three on `myDocument` if the shape is WordArt.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.FontName = "Courier New"
    End If
End With
```

FontSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproFontSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproFontSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproFontSizeA"}

Returns or sets the font size for the specified WordArt, in points. Read/write **Single**.

FontSize Property Example

This example sets the font size to 16 points for the shape named "WordArt 4" in `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes("WordArt 4").TextEffect.FontSize = 16
```

KernedPairs Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproKernedPairsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproKernedPairsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproKernedPairsA"}

True if character pairs in the specified WordArt are kerned. Read/write **Long**.

KernedPairs Property Example

This example turns on character pair kerning for shape three on `myDocument` if the shape is WordArt.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(3)
    If .Type = msoTextEffect Then
        .TextEffect.KernedPairs = True
    End If
End With
```

NormalizedHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro NormalizedHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro NormalizedHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro NormalizedHeightA"}

True if all characters (both uppercase and lowercase) in the specified WordArt are the same height. Read/write **Long**.

NormalizedHeight Property 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 = ActivePresentation.Slides(1)
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 = True
```

PresetShape Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetShapeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetShapeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetShapeA"}

Returns or sets the shape of the specified WordArt. Read/write **Long**.

Can be one of the following **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

Remarks

Setting the **PresetTextEffect** property automatically sets the **PresetShape** property.

PresetShape Property Example

This example sets the shape of all WordArt on `myDocument` to a chevron whose center points down.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.Type = msoTextEffect Then
        s.TextEffect.PresetShape = msoTextEffectShapeChevronDown
    End If
Next
```

PresetTextEffect Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetTextEffectC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetTextEffectX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetTextEffectA"}

Returns or sets the style of the specified WordArt. The values for this property correspond to the formats in the **WordArt Gallery** dialog box (numbered from left to right, top to bottom). Read/write **Long**.

Can be one of the following **MsoPresetTextEffect** constants:

msoTextEffect1	**msoTextEffect17**
msoTextEffect2	**msoTextEffect18**
msoTextEffect3	**msoTextEffect19**
msoTextEffect4	**msoTextEffect20**
msoTextEffect5	**msoTextEffect21**
msoTextEffect6	**msoTextEffect22**
msoTextEffect7	**msoTextEffect23**
msoTextEffect8	**msoTextEffect24**
msoTextEffect9	**msoTextEffect25**
msoTextEffect10	**msoTextEffect26**
msoTextEffect11	**msoTextEffect27**
msoTextEffect12	**msoTextEffect28**
msoTextEffect13	**msoTextEffect29**
msoTextEffect14	**msoTextEffect30**
msoTextEffect15	**msoTextEffectMixed**
msoTextEffect16	

Remarks

Setting the **PresetTextEffect** property automatically sets many other formatting properties of the specified shape.

PresetTextEffect Property Example

This example sets the style for all WordArt on `myDocument` to the first style listed in the **WordArt Gallery** dialog box.

```
Set myDocument = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    If s.Type = msoTextEffect Then
        s.TextEffect.PresetTextEffect = msoTextEffect1
    End If
Next
```

RotatedChars Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRotatedCharsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRotatedCharsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRotatedCharsA"}

True if characters in the specified WordArt are rotated 90 degrees relative to the WordArt's bounding shape. **False** if characters in the specified WordArt retain their original orientation relative to the bounding shape. Read/write **Long**.

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.

RotatedChars Property Example

This example adds WordArt that contains the text "Test" to `myDocument` and rotates the characters 90 degrees counterclockwise.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

ToggleVerticalText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthToggleVerticalTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthToggleVerticalTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthToggleVerticalTextA"}

Switches the text flow in the specified WordArt from horizontal to vertical, or vice versa.

Syntax

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.

ToggleVerticalText Method Example

This example adds WordArt that contains the text "Test" to `myDocument` and switches from horizontal text flow (the default for the specified WordArt style, **msoTextEffect1**) to vertical text flow.

```
Set myDocument = ActivePresentation.Slides(1)
Set newWordArt =
myDocument.Shapes.AddTextEffect(PresetTextEffect:=msoTextEffect1,
Text:="Test", _
    FontName:="Arial Black", FontSize:=36, FontBold:=False,
FontItalic:=False, Left:=100, Top:=100)
newWordArt.TextEffect.ToggleVerticalText
```

Tracking Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproTrackingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproTrackingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproTrackingA"}

Returns or sets the ratio of the horizontal space allotted to each character in the specified WordArt to the width of the character. Can be a value from 0 (zero) thorugh 5. (Large values for this property specify ample space between characters; values less than 1 can produce character overlap.) Read/write **Single**.

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

Tracking Property Example

This example adds WordArt that contains the text "Test" to `myDocument` and specifies that the characters be very tightly spaced.

```
Set myDocument = ActivePresentation.Slides(1)
Set newWordArt =
myDocument.Shapes.AddTextEffect(PresetTextEffect:=msoTextEffect1,
Text:="Test", _
    FontName:="Arial Black", FontSize:=36, FontBold:=False,
FontItalic:=False, Left:=100, Top:=100)
newWordArt.TextEffect.Tracking =0.8
```

AutoSize Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAutoSizeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAutoSizeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAutoSizeA"}

Returns or sets a value that indicates whether the size of the specified shape is changed automatically to fit text within its boundaries. Can be one of the following **PpAutoSize** constants: **ppAutoSizeMixed**, **ppAutoSizeNone**, or **ppAutoSizeShapeToFitText**. Read/write **Long**.

AutoSize Property Example

This example adjusts the size of the title bounding box on slide one to fit the title text.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1)
    If .TextFrame.TextRange.Characters.Count < 50 Then
        .TextFrame.AutoSize = ppAutoSizeShapeToFitText
    End If
End With
```

DeleteText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDeleteTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDeleteTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDeleteTextA"}

Deletes the text associated with the specified shape.

Syntax

expression.**DeleteText**

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

DeleteText Method Example

If shape two on `myDocument` contains text, this example deletes the text.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(2).TextFrame.DeleteText
```

HasText Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHasTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHasTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHasTextA"}

True if the specified shape has text associated with it. Read-only **Long**.

HasText Property Example

If shape two on `myDocument` contains text, this example resizes the shape to fit the text.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(2).TextFrame
    If .HasText Then .AutoSize = ppAutoSizeShapeToFitText
End With
```

HorizontalAnchor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproHorizontalAnchorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproHorizontalAnchorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproHorizontalAnchorA"}

Returns or sets the horizontal anchor type for the specified text. Can be one of the following **MsoHorizontalAnchor** constants: **msoAnchorCenter**, **msoAnchorNone**, or **msoHorizontalAnchorMixed**. Read/write **Long**.

HorizontalAnchor Property Example

This example sets the anchor type for text in shape one on `myDocument` to top centered.

```
Set myDocument = ActivePresentation.SlideMaster
With myDocument.Shapes(1)
    .TextFrame.HorizontalAnchor = msoAnchorCenter
    .TextFrame.VerticalAnchor = msoAnchorTop
End With
```

MarginBottom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro MarginBottomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro MarginBottomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro MarginBottomA"}

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**.

MarginBottom Property Example

This example adds a rectangle to `myDocument`, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproMarginLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproMarginLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproMarginLeftA"}

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**.

MarginLeft Property Example

This example adds a rectangle to `myDocument`, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro MarginRightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro MarginRightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro MarginRightA"}

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**.

MarginRight Property Example

This example adds a rectangle to `myDocument`, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproMarginTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproMarginTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproMarginTopA"}

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**.

MarginTop Property Example

This example adds a rectangle to `myDocument`, adds text to the rectangle, and then sets the margins for the text frame.

```
Set myDocument = ActivePresentation.Slides(1)
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
```

Orientation Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproOrientationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproOrientationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproOrientationA"}

Returns or sets text orientation. Can be one of the following **MsoTextOrientation** constants: **msoTextOrientationHorizontal**, **msoTextOrientationMixed**, or **msoTextOrientationVertical**. Read/write **Long**.

Remarks

The following **MsoTextOrientation** constants aren't used in the U.S. English versions of Microsoft Office: **msoTextOrientationDownward**, **msoTextOrientationHorizontalRotatedFarEast**, **msoTextOrientationUpward**, or **msoTextOrientationVerticalFarEast**.

Orientation Property Example

This example orients the text horizontally within shape three on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(3).TextFrame.Orientation = msoTextOrientationHorizontal
```

Ruler Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRulerC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRulerX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRulerA"}

Returns a **Ruler** object that represents the ruler for the specified text. Read-only.

Ruler Property Example

This example sets a left-aligned tab stop at 2 inches (144 points) for the text in object two on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(2).TextFrame.Ruler.TabStops.Add ppTabStopLeft, 144
```

VerticalAnchor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproVerticalAnchorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproVerticalAnchorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproVerticalAnchorA"}

Returns or sets the vertical anchor type for the specified text. Can be one of the following **MsoVerticalAnchor** constants: **msoAnchorBottom**, **msoAnchorBottomBaseLine**, **msoAnchorMiddle**, **msoAnchorTop**, **msoAnchorTopBaseline**, or **msoVerticalAnchorMixed**. Read/write **Long**.

VerticalAnchor Property Example

This example sets the anchor type for the shape one on `myDocument` to top centered.

```
Set myDocument = ActivePresentation.SlideMaster
With myDocument.Shapes(1)
    .TextFrame.HorizontalAnchor = msoAnchorCenter
    .TextFrame.VerticalAnchor = msoAnchorTop
End With
```

WordWrap Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproWordWrapC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproWordWrapX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproWordWrapA"}

TextFrame object: **True** if lines break automatically to fit inside the shape. Read/write **Long**.

ParagraphFormat object: Used only with Kanji characters. Read/write **Long**.

WordWrap Property Example

This example adds a rectangle that contains text to `myDocument` and then turns off word wrapping in the new rectangle.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeRectangle, 0, 0, 100,
300).TextFrame
    .TextRange.Text = "Here is some test text that is too long for this
box"
    .WordWrap = False
End With
```

AddPeriods Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthAddPeriodsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthAddPeriodsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthAddPeriodsA"}

Adds a period at the end of each paragraph in the specified text.

Syntax

expression.**AddPeriods**

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

Remarks

This method doesn't add another period at the end of a paragraph that already ends with a period.

AddPeriods Method Example

This example adds a period at the end of each paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame _
    .TextRange.AddPeriods
```

BoundHeight Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro BoundHeightC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro BoundHeightX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro BoundHeightA"}

Returns the height (in points) of the text bounding box for the specified text frame. Read-only **Single**.

bounding box

The smallest rectangle that can enclose the text in a text frame. Knowing the position and dimensions of text bounding boxes allows you to overlap shapes on a slide without covering up any text.

BoundHeight Property Example

This example adds a rounded rectangle to slide one in the active presentation. The rectangle has the same dimensions as the text bounding box for shape one.

```
With Application.ActivePresentation.Slides(1).Shapes
    Set tr = .Item(1).TextFrame.TextRange
    Set roundRect = .AddShape(msoShapeRoundedRectangle, _
        tr.BoundLeft, tr.BoundTop, tr.BoundWidth, tr.BoundHeight)
End With
With roundRect.Fill
    .ForeColor.RGB = RGB(255, 0, 128)
    .Transparency = 0.75
End With
```

BoundLeft Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBoundLeftC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBoundLeftX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBoundLeftA"}

Returns the distance (in points) from the left edge of the text bounding box for the specified text frame to the left edge of the slide. Read-only **Single**.

BoundLeft Property Example

This example adds a rounded rectangle to slide one in the active presentation. The rectangle has the same dimensions as the text bounding box for shape one.

```
With Application.ActivePresentation.Slides(1).Shapes
    Set tr = .Item(1).TextFrame.TextRange
    Set roundRect = .AddShape(msoShapeRoundedRectangle, _
        tr.BoundLeft, tr.BoundTop, tr.BoundWidth, tr.BoundHeight)
End With
With roundRect.Fill
    .ForeColor.RGB = RGB(255, 0, 128)
    .Transparency = 0.75
End With
```

BoundTop Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBoundTopC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBoundTopX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBoundTopA"}

Returns the distance (in points) from the top of the of the text bounding box for the specified text frame to the top of the slide. Read-only **Single**.

BoundTop Property Example

This example adds a rounded rectangle to slide one in the active presentation. The rectangle has the same dimensions as the text bounding box for shape one.

```
With Application.ActivePresentation.Slides(1).Shapes
    Set tr = .Item(1).TextFrame.TextRange
    Set roundRect = .AddShape(msoShapeRoundedRectangle, _
        tr.BoundLeft, tr.BoundTop, tr.BoundWidth, tr.BoundHeight)
End With
With roundRect.Fill
    .ForeColor.RGB = RGB(255, 0, 128)
    .Transparency = 0.75
End With
```

BoundWidth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproBoundWidthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproBoundWidthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproBoundWidthA"}

Returns the width (in points) of the text bounding box for the specified text frame. Read-only **Single**.

BoundWidth Property Example

This example adds a rounded rectangle to slide one in the active presentation. The rectangle has the same dimensions as the text bounding box for shape one.

```
With Application.ActivePresentation.Slides(1).Shapes
    Set tr = .Item(1).TextFrame.TextRange
    Set roundRect = .AddShape(msoShapeRoundedRectangle, _
        tr.BoundLeft, tr.BoundTop, tr.BoundWidth, tr.BoundHeight)
End With
With roundRect.Fill
    .ForeColor.RGB = RGB(255, 0, 128)
    .Transparency = 0.75
End With
```

ChangeCase Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthChangeCaseC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthChangeCaseX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthChangeCaseA"}

Changes the case of the specified text.

Syntax

expression.**ChangeCase(*Type*)**

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

Type Required **Long**. Specifies the way the case will be changed. Can be one of the following **PpChangeCase** constants: **ppCaseLower**, **ppCaseSentence**, **ppCaseTitle**, **ppCaseToggle**.

ChangeCase Method Example

This example sets title case capitalization for the title on slide one in the the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes.Title.TextFrame _
    .TextRange.ChangeCase ppCaseTitle
```

Characters Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthCharactersC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthCharactersX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthCharactersA"}

Returns a **TextRange** object that represents the specified subset of text characters. For information about counting or looping through the characters in a text range, see the **TextRange** object.

Syntax

expression.**Characters**(***Start, Length***)

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

Start Optional **Long**. The first character in the returned range.

Length Optional **Long**. The number of characters to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first character and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one character.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first character in the specified range.

If ***Start*** is greater than the number of characters in the specified text, the returned range starts with the last character in the specified range.

If ***Length*** is greater than the number of characters from the specified starting character to the end of the text, the returned range contains all those characters.

Characters Method Example

This example sets the text for shape two on slide one in the active presentation and then makes the second character a subscript character with a 20-percent offset.

```
Dim charRange As TextRange
With Application.ActivePresentation.Slides(1).Shapes(2)
    Set charRange = .TextFrame.TextRange.InsertBefore("H2O")
    charRange.Characters(2).Font.BaselineOffset = -0.2
End With
```

This example formats every subscript character in shape two on slide one as bold.

```
With Application.ActivePresentation.Slides(1).Shapes(2) _
    .TextFrame.TextRange
    For i = 1 To .Characters.Count
        With .Characters(i).Font
            If .Subscript Then .Bold = True
        End With
    Next
End With
```

Find Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFindC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFindX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFindA"}

Finds the specified text in a text range, and returns a **TextRange** object that represents the first text range where the text is found. Returns **Nothing** if no match is found.

Syntax

expression.**Find(*FindWhat, After, MatchCase, WholeWords*)**

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

FindWhat Required **String**. The text to search for.

After Optional **Long**. The position of the character (in the specified text range) after which you want to search for the next occurrence of ***FindWhat***. For example, if you want to search from the fifth character of the text range, specify 4 for ***After***. If this argument is omitted, the first character of the text range is used as the starting point for the search.

MatchCase Optional **Long**. **True** to have the search distinguish between uppercase and lowercase characters. The default value is **False**.

WholeWords Optional **Long**. **True** to have the search find only whole words, and not parts of larger words as well. The default value is **False**.

Find Method Example

This example finds every occurrence of "CompanyX" in the active presentation and formats it as bold.

```
For Each sld In Application.ActivePresentation.Slides
    For Each shp In sld.Shapes
        If shp.HasTextFrame Then
            Set txtRng = shp.TextFrame.TextRange
            Set foundText = txtRng.Find(FindWhat:="CompanyX")
            Do While Not (foundText Is Nothing)
                With foundText
                    .Font.Bold = True
                    Set foundText = txtRng.Find(FindWhat:="CompanyX", _
                        After:=.Start + .Length - 1)
                End With
            Loop
        End If
    Next
Next
```

IndentLevel Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproIndentLevelC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproIndentLevelX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproIndentLevelA"}

Returns or sets the the indent level for the specified text as an integer from 1 to 5, where 1 indicates a first-level paragraph with no indentation. Read/write **Long**.

IndentLevel Property Example

This example indents the second paragraph in shape two on slide two in the active presentation.

```
Application.ActivePresentation.Slides(2).Shapes(2).TextFrame _
    .TextRange .Paragraphs(2).IndentLevel = 2
```

InsertAfter Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertAfterC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertAfterX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertAfterA"}

Appends a string to the end of the specified text range. Returns a **TextRange** object that represents the appended text. When used without an argument, this method returns a zero-length string at the end of the specified range.

Syntax

expression.**InsertAfter**(***NewText***)

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

NewText Optional **String**. The text to be inserted. The default value is an empty string.

InsertAfter Method Example

This example appends the string ": Test version" to the end of the title on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(1)
    .TextFrame.TextRange.InsertAfter ": Test version"
End With
```

This example appends the contents of the Clipboard to the end of the title on slide one.

```
Application.ActivePresentation.Slides(1).Shapes(1).TextFrame _
    .TextRange.InsertAfter.Paste
```

InsertBefore Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertBeforeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertBeforeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertBeforeA"}

Appends a string to the beginning of the specified text range. Returns a **TextRange** object that represents the appended text. When used without an argument, this method returns a zero-length string at the end of the specified range.

Syntax

expression.**InsertBefore(*NewText*)**

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

NewText Optional **String**. The text to be appended. The default value is an empty string.

InsertBefore Method Example

This example appends the string "Test version: " to the beginning of the title on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(1).Shapes(1)
    .TextFrame.TextRange.InsertBefore "Test version: "
End With
```

This example appends the contents of the Clipboard to the beginning of the title on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(1).TextFrame _
    .TextRange.InsertBefore .Paste
```

InsertDateTime Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertDateTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertDateTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertDateTimeA"}

Inserts the date and time in the specified text range. Returns a **TextRange** object that represents the inserted text.

Syntax

expression.**InsertDateTime(*DateTimeFormat, InsertAsField*)**

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

DateTimeFormat Required **Long**. A format for the date and time. Can be one of the following **PpDateTimeFormat** constants:

ppDateTimeddddMMMMddyyyy	**ppDateTimeMdyy**
ppDateTimedMMMMyyyy	**ppDateTimeMMddyyHmm**
ppDateTimedMMMyy	**ppDateTimeMMddyyhmmAMPM**
ppDateTimeHmm	**ppDateTimeMMMMdyyyy**
ppDateTimehmmAMPM	**ppDateTimeMMMMyy**
ppDateTimeHmmss	**ppDateTimeMMyy**
ppDateTimehmmssAMPM	

InsertAsField Optional **Long**. **True** to have the inserted date and time be updated each time the presentation is opened. The default value is **False**.

InsertDateTime Method Example

This example inserts the date and time after the first sentence of the first paragraph in shape two on slide one in the active presentation.

```
Set sh = Application.ActivePresentation.Slides(1).Shapes(2)
Set sentOne = sh.TextFrame.TextRange.Paragraphs(1).Sentences(1)
sentOne.InsertAfter.InsertDateTime ppDateTimeMdyy
```

InsertSlideNumber Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertSlideNumberC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertSlideNumberX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertSlideNumberA"}

Inserts the slide number of the current slide into the specified text range. Returns a **TextRange** object that represents the slide number.

Syntax

expression.**InsertSlideNumber**

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

Remarks

The inserted slide number is automatically updated when the slide number of the current slide changes.

InsertSlideNumber Method Example

This example inserts the slide number of the current slide after the first sentence of the first paragraph in shape two on slide one in the active presentation.

```
Set sh = Application.ActivePresentation.Slides(1).Shapes(2)
Set sentOne = sh.TextFrame.TextRange.Paragraphs(1).Sentences(1)
sentOne.InsertAfter.InsertSlideNumber
```

InsertSymbol Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthInsertSymbolC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthInsertSymbolX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthInsertSymbolA"}

Inserts the symbol in the specified text range. Returns a **TextRange** object that represents the symbol.

Syntax

expression.**InsertSymbol(*FontName, CharNumber, UniCode*)**

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

FontName Required **String**. The font name.

CharNumber Required **Long**. The Unicode or ASCII character number.

UniCode Optional **Long**. **True** to have the ***CharNumber*** argument represent a UniCode character. **False** to have the ***CharNumber*** argument represent an ASCII character number. The default value is **False**.

InsertSymbol Method Example

This example inserts the Registered Trademark symbol after the first sentence of the first paragraph in shape two on slide one in the active presentation.

```
Set sh = Application.ActivePresentation.Slides(1).Shapes(2)
Set sentOne = sh.TextFrame.TextRange.Paragraphs(1).Sentences(1)
sentOne.InsertAfter.InsertSymbol "Symbol", 226
```

Lines Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthLinesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthLinesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthLinesA"}

Returns a **TextRange** object that represents the specified subset of text lines. For information about counting or looping through the lines in a text range, see the **TextRange** object.

Syntax

expression.**Lines(*Start, Length*)**

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

Start Optional **Long**. The first line in the returned range.

Length Optional **Long**. The number of lines to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first line and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one line.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first line in the specified range.

If ***Start*** is greater than the number of lines in the specified text, the returned range starts with the last line in the specified range.

If ***Length*** is greater than the number of lines from the specified starting line to the end of the text, the returned range contains all those lines.

Lines Method Example

This example formats as italic the first two lines of the second paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2) _
    .TextFrame.TextRange.Paragraphs(2).Lines(1, 2).Font.Italic = True
```

ParagraphFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproParagraphFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproParagraphFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproParagraphFormatA"}

Returns a **ParagraphFormat** object that represents paragraph formatting for the specified text. Read-only.

ParagraphFormat Property Example

This example sets the line spacing before, within, and after eachparagraph in shape two on slide one in the active presentation.

```
With Application.ActivePresentation.Slides(2).Shapes(2)
    With .TextFrame.TextRange.ParagraphFormat
        .LineRuleWithin = msoTrue
        .LineSpacing = 1.4
        .LineRuleBefore = msoTrue
        .SpaceBefore = 0.25
        .LineRuleAfter = msoTrue
        .SpaceAfter = 0.75
    End With
End With
```

Paragraphs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthParagraphsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthParagraphsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthParagraphsA"}

Returns a **TextRange** object that represents the specified subset of text paragraphs. For information about counting or looping through the paragraphs in a text range, see the **TextRange** object.

Syntax

expression.**Paragraphs(*Start, Length*)**

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

Start Optional **Long**. The first paragraph in the returned range.

Length Optional **Long**. The number of paragraphs to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first paragraph and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one paragraph.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first paragraph in the specified range.

If ***Start*** is greater than the number of paragraphs in the specified text, the returned range starts with the last paragraph in the specified range.

If ***Length*** is greater than the number of paragraphs from the specified starting paragraph to the end of the text, the returned range contains all those paragraphs.

Paragraphs Method Example

This example formats as italic the first two lines of the second paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2) _
    .TextFrame.TextRange.Paragraphs(2).Lines(1, 2).Font.Italic = True
```

RemovePeriods Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRemovePeriodsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRemovePeriodsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRemovePeriodsA"}

Removes the period at the end of each paragraph in the specified text.

RemovePeriods Method Example

This example removes the period at the end of each paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .RemovePeriods
```

Replace Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthReplaceC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthReplaceX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthReplaceA"}

TextRange object: Finds specific text in a text range, replaces the found text with a specified string, and returns a **TextRange** object that represents the first occurrence of the found text. Returns **Nothing** if no match is found.

Fonts collection: Replaces a font in the **Fonts** collection.

Syntax 1

expression.**Replace(*FindWhat, ReplaceWhat, After, MatchCase, WholeWords*)**

Syntax 2

expression.**Replace(*Original, Replacement*)**

expression Required. An expression that returns a **TextRange** object (Syntax 1) or a **Fonts** collection (Syntax 2).

FindWhat Required **String**. The text to search for.

ReplaceWhat Required **String**. The text you want to replace the found text with.

After Optional **Long**. The position of the character (in the specified text range) after which you want to search for the next occurrence of ***FindWhat***. For example, if you want to search from the fifth character of the text range, specify 4 for ***After***. If this argument is omitted, the first character of the text range is used as the starting point for the search.

MatchCase Optional **Long**. **True** to have the search distinguish between uppercase and lowercase characters. The default value is **False**.

WholeWords Optional **Long**. **True** to have the search find only whole words, and not parts of larger words. The default value is **False**.

Original Required **String**. The name of the font to replace.

Replacement Required **String**. The the name of the replacement font.

Replace Method Example

This example replaces every whole-word occurrence of "CompanyX" in the active presentation with "CompanyY."

```
For Each sld In Application.ActivePresentation.Slides
    For Each shp In sld.Shapes
        shp.TextFrame.TextRange.Replace FindWhat:="CompanyX", _
            Replacewhat:="CompanyY", WholeWords:=True
    Next shp
Next sld
```

This example replaces the Times New Roman font with the Courier font in the active presentation.

```
Application.ActivePresentation.Fonts.Replace "Times New Roman", "Courier"
```

Runs Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthRunsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthRunsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthRunsA"}

Returns a **TextRange** object that represents the specified subset of text runs. A text run consists of a range of characters that share the same font attributes. For information about counting or looping through the runs in a text range, see the **TextRange** object.

Syntax

expression.**Runs(*Start, Length*)**

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

Start Optional **Long**. The first run in the returned range.

Length Optional **Long**. The number of runs to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first run and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one run.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first run in the specified range.

If ***Start*** is greater than the number of runs in the specified text, the returned range starts with the last run in the specified range.

If ***Length*** is greater than the number of runs from the specified starting run to the end of the text, the returned range contains all those runs.

A run consists of all characters from the first character after a font change to the second-to-last character with the same font attributes. For example, consider the following sentence:

This *italic* word is not **bold**.

In the preceding sentence, the first run consists of the word "This" only if the space after the word "This" isn't formatted as italic (if the space is italic, the first run is only the first three characters, or "Thi"). Likewise, the second run contains the word "italic" only if the space after the word is formatted as italic.

Runs Method Example

This example formats the second run in shape two on slide one in the active presentation as bold italic if it's already italic.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange
    With .Runs(2).Font
        If .Italic Then
            .Bold = True
        End If
    End With
End With
```

Sentences Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSentencesC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSentencesX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSentencesA"}

Returns a **TextRange** object that represents the specified subset of text sentences. For information about counting or looping through the sentences in a text range, see the **TextRange** object.

Syntax

expression.**Sentences(*Start, Length*)**

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

Start Optional **Long**. The first sentence in the returned range.

Length Optional **Long**. The number of sentences to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first sentence and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one sentence.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first sentence in the specified range.

If ***Start*** is greater than the number of sentences in the specified text, the returned range starts with the last sentence in the specified range.

If ***Length*** is greater than the number of sentences from the specified starting sentence to the end of the text, the returned range contains all those sentences.

Sentences Method Example

This example formats as bold the second sentence in the second paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .Paragraphs(2).Sentences(2).Font.Bold = True
```

TrimText Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthTrimTextC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthTrimTextX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthTrimTextA"}

Returns a **TextRange** object that represents the specified text minus any trailing spaces.

Syntax

expression.**TrimText**

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

TrimText Method Example

This example inserts the string " Text to trim " at the beginning of the text in shape two on slide one in the active presentation and then displays message boxes showing the string before and after it's trimmed.

```
With Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange
    With .InsertBefore("   Text to trim   ")
        MsgBox "Untrimmed: " & """" & .Text & """"
        MsgBox "Trimmed: " & """" & .TrimText.Text & """"
    End With
End With
```

Words Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthWordsC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthWordsX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthWordsA"}

Returns a **TextRange** object that represents the specified subset of text words. For information about counting or looping through the words in a text range, see the **TextRange** object.

Syntax

expression.**Words(*Start, Length*)**

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

Start Optional **Long**. The first word in the returned range.

Length Optional **Long**. The number of words to be returned.

Remarks

If both ***Start*** and ***Length*** are omitted, the returned range starts with the first word and ends with the last paragraph in the specified range.

If ***Start*** is specified but ***Length*** is omitted, the returned range contains one word.

If ***Length*** is specified but ***Start*** is omitted, the returned range starts with the first word in the specified range.

If ***Start*** is greater than the number of words in the specified text, the returned range starts with the last word in the specified range.

If ***Length*** is greater than the number of words from the specified starting word to the end of the text, the returned range contains all those words.

Words Method Example

This example formats as bold the second, third, and fourth words in the first paragraph in shape two on slide one in the active presentation.

```
Application.ActivePresentation.Slides(1).Shapes(2).TextFrame.TextRange _
    .Paragraphs(1).Words(2, 3).Font.Bold = True
```

Depth Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDepthC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDepthX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDepthA"}

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**.

Depth Property Example

This example adds an oval to `myDocument` and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Set myDocument = ActivePresentation.Slides(1)
Set myShape = myDocument.Shapes.AddShape(msoShapeOval, 90, 90, 90, 40)
With myShape.ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)    ' RGB value for purple
End With
```

ExtrusionColor Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproExtrusionColorC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproExtrusionColorX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproExtrusionColorA"}

Returns a **ColorFormat** object that represents the color of the shape's extrusion. Read-only.

ExtrusionColor Property Example

This example adds an oval to `myDocument` and then specifies that the oval be extruded to a depth of 50 points and that the extrusion be purple.

```
Set myDocument = ActivePresentation.Slides(1)
Set myShape = myDocument.Shapes.AddShape(msoShapeOval, 90, 90, 90, 40)
With myShape.ThreeD
    .Visible = True
    .Depth = 50
    .ExtrusionColor.RGB = RGB(255, 100, 255)    ' RGB value for purple
End With
```

ExtrusionColorType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproExtrusionColorTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproExtrusionColorTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproExtrusionColorTypeA"}

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. Can be one of the following **MsoExtrusionColorType** constants: **msoExtrusionColorAutomatic** (extrusion color based on shape fill), **msoExtrusionColorCustom** (extrusion color independent of shape fill), or **msoExtrusionColorTypeMixed**. Read/write **Long**.

ExtrusionColorType Property Example

If shape one on `myDocument` has an automatic extrusion color, this example gives the extrusion a custom yellow color.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    If .ExtrusionColorType = msoExtrusionColorAutomatic Then
        .ExtrusionColor.RGB = RGB(240, 235, 16)
    End If
End With
```

IncrementRotationX Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementRotationXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementRotationXXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementRotationXA"}

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.

Syntax

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.

IncrementRotationX Method Example

This example tilts shape one on `myDocument` up 10 degrees. Shape one must be an extruded shape for you to see the effect of this code.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).ThreeD.IncrementRotationX 10
```

IncrementRotationY Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthIncrementRotationYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthIncrementRotationYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthIncrementRotationYA"}

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.

Syntax

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.

IncrementRotationY Method Example

This example tilts shape one on `myDocument` 10 degrees to the right. Shape one must be an extruded shape for you to see the effect of this code.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).ThreeD.IncrementRotationY -10
```

Perspective Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro PerspectiveC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro PerspectiveX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro PerspectiveA"}

True if the extrusion appears in perspective—that is, if the walls of the extrusion narrow toward a vanishing point. **False** if the extrusion is a parallel, or orthographic, projection—that is, if the walls don't narrow toward a vanishing point. Read/write **Long**.

Perspective Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .Depth = 100
    .Perspective = False
End With
```

PresetExtrusionDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetExtrusionDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetExtrusionDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetExtrusionDirectionA"}

Returns the direction that the extrusion's sweep path takes away from the extruded shape (the front face of the extrusion). Can be one of the following **MsoPresetExtrusionDirection** constants: **msoExtrusionBottom**, **msoExtrusionBottomLeft**, **msoExtrusionBottomRight**, **msoExtrusionLeft**, **msoExtrusionNone**, **msoExtrusionRight**, **msoExtrusionTop**, **msoExtrusionTopLeft**, **msoExtrusionTopRight**, or **msoPresetExtrusionDirectionMixed**. Read-only **Long**.

Remarks

This property is read-only. To set the value of this property, use the **SetExtrusionDirection** method.

PresetExtrusionDirection Property 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 = ActivePresentation.Slides(1)
For Each s In myDocument.Shapes
    With s.ThreeD
        If .PresetExtrusionDirection = msoExtrusionTopLeft Then
            .SetExtrusionDirection msoExtrusionBottomRight
        End If
    End With
Next
```

PresetLightingDirection Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetLightingDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetLightingDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetLightingDirectionA"}

Returns or sets the position of the light source relative to the extrusion. Can be one of the following **MsoPresetLightingDirection** constants: **msoLightingBottom**, **msoLightingBottomLeft**, **msoLightingBottomRight**, **msoLightingLeft**, **msoLightingNone**, **msoLightingRight**, **msoLightingTop**, **msoLightingTopLeft**, **msoLightingTopRight**, or **msoPresetLightingDirectionMixed**. Read/write **Long**.

Note You won't see the lighting effects you set if the extrusion has a wire frame surface.

PresetLightingDirection Property 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```

PresetLightingSoftness Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetLightingSoftnessC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetLightingSoftnessX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetLightingSoftnessA"}

Returns or sets the intensity of the extrusion lighting. Can be one of the following **MsoPresetLightingSoftness** constants: **msoLightingBright**, **msoLightingDim**, **msoLightingNormal**, or **msoPresetLightingSoftnessMixed**. Read/write **Long**.

PresetLightingSoftness Property Example

This example specifies that the extrusion for shape one on `myDocument` be lit brightly from the left.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetLightingSoftness = msoLightingBright
    .PresetLightingDirection = msoLightingLeft
End With
```

PresetMaterial Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetMaterialC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetMaterialX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetMaterialA"}

Returns or sets the extrusion surface material. Can be one of the following **MsoPresetMaterial** constants: **msoMaterialMatte**, **msoMaterialMetal**, **msoMaterialPlastic**, **msoMaterialWireFrame**, or **msoPresetMaterialMixed**. Read/write **Long**.

PresetMaterial Property Example

This example specifies that the extrusion surface for shape one in `myDocument` be wire frame.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .PresetMaterial = msoMaterialWireFrame
End With
```

PresetThreeDFormat Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresetThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresetThreeDFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresetThreeDFormatA"}

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**. Can be one of the following **MsoPresetThreeDFormat** constants: **msoPresetThreeDFormatMixed**, **msoThreeD1**, **msoThreeD10**, **msoThreeD11**, **msoThreeD12**, **msoThreeD13**, **msoThreeD14**, **msoThreeD15**, **msoThreeD16**, **msoThreeD17**, **msoThreeD18**, **msoThreeD19**, **msoThreeD2**, **msoThreeD20**, **msoThreeD3**, **msoThreeD4**, **msoThreeD5**, **msoThreeD6**, **msoThreeD7**, **msoThreeD8**, or **msoThreeD9**. 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. Read-only **Long**.

Remarks

This property is read-only. To set the preset extrusion format, use the **SetThreeDFormat** method.

PresetThreeDFormat Property Example

This example sets the extrusion format for shape one on `myDocument` to 3D Style 12 if the shape initially has a custom extrusion format.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    If .PresetThreeDFormat = msoPresetThreeDFormatMixed Then
        .SetThreeDFormat msoThreeD12
    End If
End With
```

ResetRotation Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthResetRotationC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthResetRotationX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthResetRotationA"}

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.

Syntax

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.

ResetRotation Method Example

This example resets the rotation around the x-axis and the y-axis to 0 (zero) for the extrusion of shape one on `myDocument`.

```
Set myDocument = ActivePresentation.Slides(1)
myDocument.Shapes(1).ThreeD.ResetRotation
```

RotationX Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRotationXC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRotationXXX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRotationXA"}

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**.

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.

RotationX Property Example

This example adds three identical extruded ovals to `myDocument` and sets their rotation around the x-axis to – 30, 0, and 30 degrees, respectively.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.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

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproRotationYC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproRotationYX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproRotationYA"}

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**.

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.

RotationY Property 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 = ActivePresentation.Slides(1)
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
```

SetExtrusionDirection Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetExtrusionDirectionC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetExtrusionDirectionX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetExtrusionDirectionA"}

Sets the direction that the extrusion's sweep path takes away from the extruded shape.

Syntax

expression.**SetExtrusionDirection(*PresetExtrusionDirection*)**

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

PresetExtrusionDirection Required **Long**. Specifies the extrusion direction. Can be one of the following **MsoPresetExtrusionDirection** constants: **msoExtrusionBottom**, **msoExtrusionBottomLeft**, **msoExtrusionBottomRight**, **msoExtrusionLeft**, **msoExtrusionNone**, **msoExtrusionRight**, **msoExtrusionTop**, **msoExtrusionTopLeft**, or **msoExtrusionTopRight**.

Remarks

This method sets the **PresetExtrusionDirection** property to the direction specified by the ***PresetExtrusionDirection*** argument.

SetExtrusionDirection Method 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 = ActivePresentation.Slides(1)
With myDocument.Shapes(1).ThreeD
    .Visible = True
    .SetExtrusionDirection msoExtrusionTop
    .PresetLightingDirection = msoLightingLeft
End With
```

SetThreeDFormat Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthSetThreeDFormatC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthSetThreeDFormatX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthSetThreeDFormatA"}

Sets the preset extrusion format. Each preset extrusion format contains a set of preset values for the various properties of the extrusion.

Syntax

expression.**SetThreeDFormat(*PresetThreeDFormat*)**

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

PresetThreeDFormat Required **Long**. Specifies a preset extrusion format that corresponds to one of the options (numbered from left to right, from top to bottom) displayed when you click the **3-D** button on the **Drawing** toolbar. Can be one of the following **MsoPresetThreeDFormat** constants: **msoThreeD1**, **msoThreeD10**, **msoThreeD11**, **msoThreeD12**, **msoThreeD13**, **msoThreeD14**, **msoThreeD15**, **msoThreeD16**, **msoThreeD17**, **msoThreeD18**, **msoThreeD19**, **msoThreeD2**, **msoThreeD20**, **msoThreeD3**, **msoThreeD4**, **msoThreeD5**, **msoThreeD6**, **msoThreeD7**, **msoThreeD8**, or **msoThreeD9**. Note that specifying **msoPresetThreeDFormatMixed** for this argument causes an error.

Remarks

This method sets the **PresetThreeDFormat** property to the format specified by the ***PresetThreeDFormat*** argument.

SetThreeDFormat Method Example

This example adds an oval to `myDocument` and sets its extrusion format to 3D Style 12.

```
Set myDocument = ActivePresentation.Slides(1)
With myDocument.Shapes.AddShape(msoShapeOval, 30, 30, 50, 25).ThreeD
    .Visible = True
    .SetThreeDFormat msoThreeD12
End With
```

AcceleratorsEnabled Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproAcceleratorsEnabledC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproAcceleratorsEnabledX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproAcceleratorsEnabledA"}

True if shortcut keys are enabled during a slide show. The default value is **True**. If shortcut keys are disabled during a slide show, you can neither use keys to navigate in the slide show nor press ESC to exit the slide show. Read/write **Long**.

AcceleratorsEnabled Property Example

This example runs a slide show of the active presentation with shortcut keys disabled.

```
ActivePresentation.SlideShowSettings.Run.View.AcceleratorsEnabled = False
```

DisplaySlideMiniature Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproDisplaySlideMiniatureC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproDisplaySlideMiniatureX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproDisplaySlideMiniatureA"}

True if the slide miniature window is displayed. Read/write **Long**.

Remarks

The slide miniature window isn't a member of either the **Windows** collection or the **SlideShowWindows** collection.

DisplaySlideMiniature Property Example

If document window one is in slide view, this example displays the slide miniature window.

```
With Windows(1).View
    If .Type = ppViewSlide Then .DisplaySlideMiniature = True
End With
```

DrawLine Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthDrawLineC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthDrawLineX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthDrawLineA"}

Draws a line in the specified slide show view.

Syntax

expression.**DrawLine(*BeginX, BeginY, EndX, Height*)**

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

BeginX, ***BeginY*** Required **Single**. The position (in points) of the line's starting point relative to the upper-left corner of the slide.

EndX, ***EndY*** Required **Single**. The position (in points) of the line's ending point relative to the upper-left corner of the slide.

DrawLine Method Example

This example draws a line in slide show window one.

```
SlideShowWindows(1).View.DrawLine 5, 5, 250, 250
```

EraseDrawing Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthEraseDrawingC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthEraseDrawingX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthEraseDrawingA"}

Removes lines drawn during a slide show using either the **DrawLine** method or the pen tool.

Syntax

expression.**EraseDrawing**

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

EraseDrawing Method Example

This example erases any lines that have been drawn in slide show window one using either the **DrawLine** method or the pen tool.

```
SlideShowWindows(1).View.EraseDrawing
```

Exit Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthExitC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthExitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthExitA"}

Ends the specified slide show.

Syntax

expression.**Exit**

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

Exit Method Example

This example ends the slide show that's running in slide show window one.

```
SlideShowWindows(1).View.Exit
```

First Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthFirstC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthFirstX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthFirstA"}

Sets the specified slide show view to display the first slide in the presentation.

Syntax

expression.**First**

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

Remarks

If you use the **First** method to switch from one slide to another during a slide show, when you return to the original slide, its animation picks up where it left off.

First Method Example

This example sets slide show window one to display the first slide in the presentation.

```
SlideShowWindows(1).View.First
```

GotoSlide Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthGotoSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthGotoSlideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthGotoSlideA"}

View object: Switches to the specified slide.

SlideShowView object: Switches to the specified slide during a slide show. You can specify whether you want the animation effects to be rerun.

Syntax 1

expression.**GotoSlide(*Index*)**

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

Index Required **Long**. The number of the slide to switch to.

Syntax 1

expression.**GotoSlide(*Index*, *ResetSlide*)**

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

Index Required **Long**. The number of the slide to switch to.

ResetSlide Optional **Long**. If you switch from one slide to another during a slide show with ***ResetSlide*** set to **False**, when you return to the first slide, its animation picks up where it left off. If you switch from one slide to another with ***ResetSlide*** set to **True**, when you return to the first slide, its entire animation starts over. The default value is **True**.

GotoSlide Method Example

This example switches from the current slide to the slide three in slide show window one. If you switch back to the current slide during the slide show, its entire animation will start over.

```
With SlideShowWindows(1).View
    .GotoSlide 3
End With
```

This example switches from the current slide to the slide three in slide show window one. If you switch back to the current slide during the slide show, its animation will pick up where it left off.

```
With SlideShowWindows(1).View
    .GotoSlide 3, False
End With
```

IsNamedShow Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproIsNamedShowC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproIsNamedShowX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproIsNamedShowA"}

True if a custom (named) slide show is displayed in the specified slide show view. Read-only **Long**.

IsNamedShow Property Example

If the slide show running in slide show window one is a custom slide show, this example displays its name.

```
With SlideShowWindows(1).View
    If .IsNamedShow Then
        MsgBox "Now showing in slide show window 1: " & .SlideShowName
    End If
End With
```

Last Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthLastC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthLastX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthLastA"}

Sets the specified slide show view to display the last slide in the presentation.

Syntax

expression.**Last**

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

Remarks

If you use the **Last** method to switch from one slide to another during a slide show, when you return to the original slide, its animation picks up where it left off.

Last Method Example

This example sets slide show window one to display the last slide in the presentation.

```
SlideShowWindows(1).View.Last
```

PointerType Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppproPointerTypeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppproPointerTypeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppproPointerTypeA"}

Returns or sets the type of pointer used in the slide show. Can be one of the following **PpSlideShowPointerType** constants: **ppSlideShowPointerAlwaysHidden**, **ppSlideShowPointerArrow**, **ppSlideShowPointerNone**, or **ppSlideShowPointerPen**. Read/write **Long**.

PointerType Property Example

This example runs a slide show of the active presentation, changes the pointer to a pen, and sets the pen color for this slide show to red.

```
Set currView = ActivePresentation.SlideShowSettings.Run.View
With currView
    .PointerColor.RGB = RGB(255, 0, 0)
    .PointerType = ppSlideShowPointerPen
End With
```

PresentationElapsedTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproPresentationElapsedTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproPresentationElapsedTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproPresentationElapsedTimeA"}

Returns the number of seconds that have elapsed since the beginning of the specified slide show. Read-only **Long**.

PresentationElapsedTime Property Example

This example goes to slide seven in slide show window one if more than five minutes have elapsed since the beginning of the slide show.

```
With SlideShowWindows(1).View
    If .PresentationElapsedTime > 300 Then
        .GotoSlide 7
    End If
End With
```

ResetSlideTime Method

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppmthResetSlideTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppmthResetSlideTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppmthResetSlideTimeA"}

Resets the elapsed time (represented by the **SlideElapsedTime** property) for the slide that's currently displayed to 0 (zero).

Syntax

expression.**ResetSlideTime**

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

ResetSlideTime Method Example

This example resets the elapsed time for the slide that's currently displayed in slide show window one to 0 (zero).

```
SlideShowWindows(1).View.ResetSlideTime
```

Slide Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideA"}

SlideShowView object: Returns a **Slide** object that represents the slide that's currently displayed in the specified slide show window view. Read-only.

View object: Returns or sets a **Slide** object that represents the slide that's currently displayed in the specified document window view. Read/write.

Remarks

If the currently displayed slide is from an embedded presentation, you can use the **Parent** property of the **Slide** object returned by the **Slide** property to return the embedded presentation that contains the slide. (The **Presentation** property of the **SlideShowWindow** object or **DocumentWindow** object returns the presentation from which the window was created, not the embedded presentation.)

Slide Property Example

This example places on the Clipboard a copy of the slide that's currently displayed in slide show window one.

```
SlideShowWindows(1).View.Slide.Copy
```

This example displays the name of the presentation currently running in slide show window one.

```
MsgBox SlideShowWindows(1).View.Slide.Parent.Name
```

SlideElapsedTime Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro SlideElapsedTimeC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro SlideElapsedTimeX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro SlideElapsedTimeA"}

Returns the number of seconds that the current slide has been displayed. Read/write **Long**.

Remarks

Use the **ResetSlideTime** method to reset the elapsed time for the slide that's currently displayed.

SlideElapsedTime Property Example

This example sets a variable to the elapsed time for the slide that's currently displayed in slide show window one and then displays the value of the variable.

```
currTime = SlideShowWindows(1).View.SlideElapsedTime
MsgBox currTime
```

SlideShowName Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproSlideShowNameC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproSlideShowNameX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproSlideShowNameA"}

ActionSetting object: Returns or sets the name of the custom slide show to run in response to a mouse action on the shape during a slide show. Read/write **String**.

PrintOptions object: Returns or sets the name of the custom slide show to print. Read/write **String**.

SlideShowView object: Returns the name of the custom slide show that's currently running in the specified slide show view. Read-only **String**.

Remarks

The **RangeType** property must be set to **ppPrintNamedSlideShow** to print a custom slide show.

SlideShowName Property Example

If the slide show running in slide show window one is a custom slide show, this example displays its name.

```
With SlideShowWindows(1).View
    If .IsNamedShow Then
        MsgBox "Now showing in slide show window 1: " & .SlideShowName
    End If
End With
```

This example prints the custom slide show "Technical Talk".

```
With ActivePresentation.PrintOptions
    .RangeType = ppPrintNamedSlideShow
    .SlideShowName = "tech talk"
End With
ActivePresentation.PrintOut
```

State Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"pppro StateC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"pppro StateX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"pppro StateA"}

Returns or sets the state of the slide show. Can be one of the following **PpSlideShowState** constants: **ppSlideShowBlackScreen**, **ppSlideShowDone**, **ppSlideShowPaused**, **ppSlideShowRunning**, **ppSlideShowWhiteScreen**. Read/write **Long**.

State Property Example

This example sets the view state in slide show window one to a black screen.

```
SlideShowWindows(1).View.State = ppSlideShowBlackScreen
```

Zoom Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproZoomC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproZoomX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproZoomA"}

SlideShowView object: Returns the zoom setting of the specified slide show window view as a percentage of normal size. Can be a value from 10 to 400. Read-only **Long**.

View object: Returns or sets the zoom setting of the specified view as a percentage of normal size. Can be a value from 10 to 400. Read/write **Long**.

Zoom Property Example

The following example sets the zoom to 30 percent for the view in document window one.

```
Windows(1).View.Zoom = 30
```

ZoomToFit Property

{ewc HLP95EN.DLL, DYNALINK, "See Also":"ppproZoomToFitC"} {ewc HLP95EN.DLL, DYNALINK, "Example":"ppproZoomToFitX":1} {ewc HLP95EN.DLL, DYNALINK, "Applies To":"ppproZoomToFitA"}

True if the view is zoomed to fit the dimensions of the document window every time the document window is resized. This property applies only to slide view, notes page view, or master view. Read/write **Long**.

Remarks

When the value of the **Zoom** property is explicity set, the value of the **ZoomToFit** property is automatically set to **False**.

ZoomToFit Property Example

The following example sets the view in document window one to slide view, with the zoom automatically set to fit the dimensions of the window.

```
With Windows(1)
    .ViewType = ppViewSlide
    .View.ZoomToFit = True
End With
```